UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria Biomèdica

CLASSIFICACIÓ AUTOMÀTICA DE PLANS ECOGRÀFICS EN ECOGRAFIES PRENATALS

Memòria i Annexos

Autor: Vicent Pérez Gregori
Director: Raúl Benítez Iglesias

Convocatòria: Juny 2018

Resum

La Intel·ligència Artificial ha anat progressant al llarg de les últimes dècades a un ritme vertiginós, que ha permès entre altres àrees, un gran avanç en l'aprenentatge automàtic (més conegut pel seu nom en anglès com Machine Learning). Un dels principals camps que s'han beneficiat d'aquests progressos tecnològics ha sigut la Visió per Computador, que té en l'Àmbit d'Imatges Mèdiques una gran quantitat de reptes emocionants. La utilització d'aquest tipus de tecnologies ha esdevingut en un augment considerable de les eines de que disposen els professionals mèdics a l'hora de prendre millors decisions sobre la salut dels pacients. En particular, l'àrea de la Imatge Mèdica que té un paper més important en l'atenció prenatal és l'ecografia obstetrícia, mètode que proporciona un diagnòstic lo més precís possible de les anomalies estructurals fetals amb l'objectiu d'optimitzar l'evolució en el període de la gestació i perinatal. Per tant, el desenvolupament d'una eina amb la capacitat d'identificar automàticament els patrons d'aquests talls biomètrics esdevé una necessitat.

Aquesta tesi pretén trobar un mètode que permeti classificar automàticament els diferents Talls Biomètrics Fetals Estàndard realitzats a la rutina de l'Ecografia Obstètrica del Primer Trimestre d'embaràs. L'estratègia proposada es basa en l'aplicació de tècniques d'aprenentatge supervisat, amb la utilització de CNN (Xarxes Neuronals de Convolució), mitjançant dues metodologies. La primera alternativa consisteix en la utilització d'una CNN com a Extractor de Característiques i seguit d'un classificador SVM (Support Vector Machine). La segona en canvi consisteix en la utilització d'una CNN capaç de realitzar tot aquest procés. Finalment, una avaluació comparativa de les prestacions que ofereixen cadascuna d'aquestes alternatives determina quina alternativa és la més adient.

Paraules clau: Xarxes Neuronals de Convolució, Pla Biomètric Fetal, Extractor de Característiques, Estructures Anatòmiques, Transferència de l'Aprenentatge, Imatge Ecogràfica, Capa.

Resumen

La Inteligencia Artificial ha ido progresando a lo largo de las últimas décadas a un ritmo vertiginoso, que ha permitido un gran avance en el aprendizaje automático (más conocido por su nombre en inglés como Machine Learning). Uno de los principales campos que se han beneficiado de estos progresos tecnológicos ha sido la Visión por Computador, que tiene en el ámbito de las Imágenes Médicas una gran cantidad de retos emocionantes. La utilización de estos tipos de tecnologías ha propiciado un aumento considerable de las herramientas de que disponen los profesionales médicos para tomar mejores decisiones sobre la salud de los pacientes. En particular, el área de la Imagen Médica que tiene un papel más importante en la atención prenatal es la ecografía obstetricia, método que proporciona un diagnóstico lo más preciso posible de las anomalías estructurales fetales con el objetivo de optimizar la evolución en el período de gestación y perinatal. Por tanto, el desarrollo de una herramienta con la capacidad de identificar automáticamente los patrones de los cortes biométricos acontece una necesidad.

Este proyecto pretende encontrar un método que permita clasificar automáticamente los diferentes Cortes Biométricos Fetales Estándar realizados durante la rutina de la Ecografía Obstétrica del Primer Trimestre de embarazo. La estrategia propuesta se basa en la aplicación de técnicas de aprendizaje supervisado, a través de la utilización de CNN (Redes Neuronales de Convolución), mediante dos metodologías. La primera alternativa consiste en la utilización de una CNN como Extractor de Características seguido de un clasificador SVM (Support Vector Machine). La segunda en cambio consiste en la utilización de una CNN capaz de realizar todo este proceso. Finalmente, una evaluación comparativa de las prestaciones que ofrece cada una de las alternativas determina cual es la más adecuada.

Palabras clave: Redes Neuronales de Convolución, Cortes Biométricos Fetales, Extractor de Características, Estructuras Anatómicas, Transferencia del Aprendizaje, Imagen Ecográfica, Capa.

Abstract

Artificial Intelligence has been progressing during the last decades with such a vertiginous rate that has allowed a huge breakthrough in the field of Machine Learning. One field that has taken advantage of that is Computer Vision, which has in the sphere of Medical Images a large quantity of exciting challenges. Using that kind of technologies has meant a considerable rising of the available tools that doctors can use in order to make better decisions about the health of their patients. The field of Medical Imaging that has the most importance in prenatal care is Fetal Ultrasound screening, a method that provides a diagnosis as accurate as possible of fetal structural anomalies with the aim of optimizing the pregnancy period and perinatal. Therefore, the development of a tool with the ability to automatically identify Fetal Biometric Patterns becomes a necessity.

This thesis aims to find a method that allows to automatically classify the Fetal Biometric Standard Planes performed in the First Trimester Ultrasound examination routine. The proposed strategy is based on the application of supervised learning techniques using CNN (Convolutional Neural Networks), through two methodologies. The first alternative relies on using a CNN as Feature Extractor followed by a SVM (Support Vector Machine) classifier. The second consist in using a CNN that is able to make all the process. Finally, a comparative evaluation of the performance offered by each of the alternatives determine which alternative is the most appropriate.

Keywords: Convolutional Neural Network, Fetal Biometric Plane, Feature Extractor, Anatomical Structure, Transfer Learning, Ultrasound Image, Layer.

Agraïments

Estic molt agraït al que ha sigut el meu tutor del Treball de Fi de Grau el Prof. Raúl Benítez per tot el que m'ha transmès al llarg d'aquests mesos, per tot el que m'ha ensenyat i pels seus bons consells a l'hora de com enfocar moltes parts del projecte.

Agrair al doctor Joan Sabrià, responsable de la Unitat de Diagnòstic prenatal de l'Hospital Sant Joan de Déu de Barcelona per presentar-nos el problema que ha esdevingut en la realització d'aquest projecte, i per la seva contribució al facilitar-nos les imatges d'ecogràfiques que han constituït el nostre conjunt de dades.

Per últim, agrair als companys del LASSIE LAB, Clàudia Serrano, Carme Nolla i Xavier Marimón el seu suport i comprensió sempre que l'he necessitat durant el desenvolupament del projecte. I per últim, a la meva família per aconseguir que haja arribat fins aquí i no desistir mai en l'empenta que els caracteritza.

Glossari

BPD **:**Diàmetre Biparietal

CNN **:** Xarxa Neuronal de Convolució

CRL **:** Longitud Crani-caudal

NT **:** Translucidesa Nucal

SVM **:** Support Vector Machine

Índex

1. Introducció

1.1. Ecografia fetal de Primer trimestre

L'ecografia és el millor mètode de screening prenatal disponible. I el principal objectiu del seu ús es comprovar la viabilitat del fetus i el seguiment del seu desenvolupament. És el mètode més segur per a conèixer l'estat general de l'embaràs. Encara que la realització de l'ecografia clínica rutinària en mode B (representació bidimensional dels ecos reflexats en forma de punts lluminosos de claredat variable segons la freqüència utilitzada i la profunditat) i en temps real és segura en tots els trimestres de l'embaràs, es recomanable evitar el seu ús rutinari abans de les 11 setmanes. Això, es degut a que tant el Doppler polsat com el color utilitzen altes potències, on els possibles efectes tèrmics i mecànics derivats del seu ús podrien resultar perjudicials per al fetus abans d'aquesta data.

Els defectes congènits són els majors responsables de la morbimortalitat perinatal i infantil junt a l'anòxia i la prematuritat. Encara que també en molts altres casos que no impliquen la mort, són la causa de complicacions i de seqüeles associades a diferents graus de discapacitat que comprometen el desenvolupament i la integració social de l'individu. Per aquest motiu, pren tanta importància la realització de les diferents sessions ecogràfiques durant el període de gestació. Aquestes, són un total de 3 sessions ecogràfiques repartides entre el primer, segon i tercer trimestre.

1.1.1. Objectiu de de les Sessions d'Ecografia Fetal

En el present projecte sols s'analitzaran aspectes referents a sessió realitzada durant el primer trimestre de gestació.
Els objectius específics de l'ecografia obstètrica que es realitza durant el primer trimestre de la gestació són els següents:

- Confirmació de la viabilitat fetal
- Determinació del nombre de fetus i, en cas de gestació múltiple, determinació de la corionicitat i l'amnicitat.
- Datació de la gestació.
- Estudi anatòmic fetal precoç
- Mesura de la translucidesa nucal
- Cribratge de patologia uterina i annexal

L'exploració referent al primer trimestre s'ha de realitzar entre la setmana 11 i la 13,6 de gestació (preferentment la setmana 12), que és quan l'embrió té una mesura de la longitud crani-caudal (LCC) o crown-rump-length (CRL) de 45 a 84 mm, segons el "Protocol de diagnòstic prenatal d'anomalies congènites fetals" vigent, pautes que coincideixen amb les de la ISUOG (International Society of Ultrasound in Obstetrics and Gynecology) i la SEGO (Sociedad Española de Ginecología y Obstetricia).

Entre tots els diferents plans d'una sessió ecogràfica a un fetus, en aquesta secció es tractaran aquells que permeten obtenir la mesura del diàmetre biparietal (BPD), la mesura de la distància crani-caudal (CLR) i el mesurament de la transparència nucal.

- **Diàmetre Biparietal – BPD:**

En el moment de la realització de la primera sessió ecogràfica, el crani s'hi troba pràcticament ocupat pels ventricles laterals, que contenen els plexes coroidals, visualitzant-se clarament l'ossificació i la integritat del cap del fetus. Si la correcta presa de la imatge ha sigut correcta, s'ha de visualitzar en talls axials que els hemisferis siguin simètrics i clarament separats per la línia interhemisfèrica. També, tant la integritat i morfologia de la calota, com l'ossificació del crani (que s'ha de poder visualitzar al final de la setmana 11) i els ventricles laterals i plexes coroidals. En la Figura 1-1 s'hi observa una mostra del que seria un Standard del Pla del Diàmetre Biparietal.

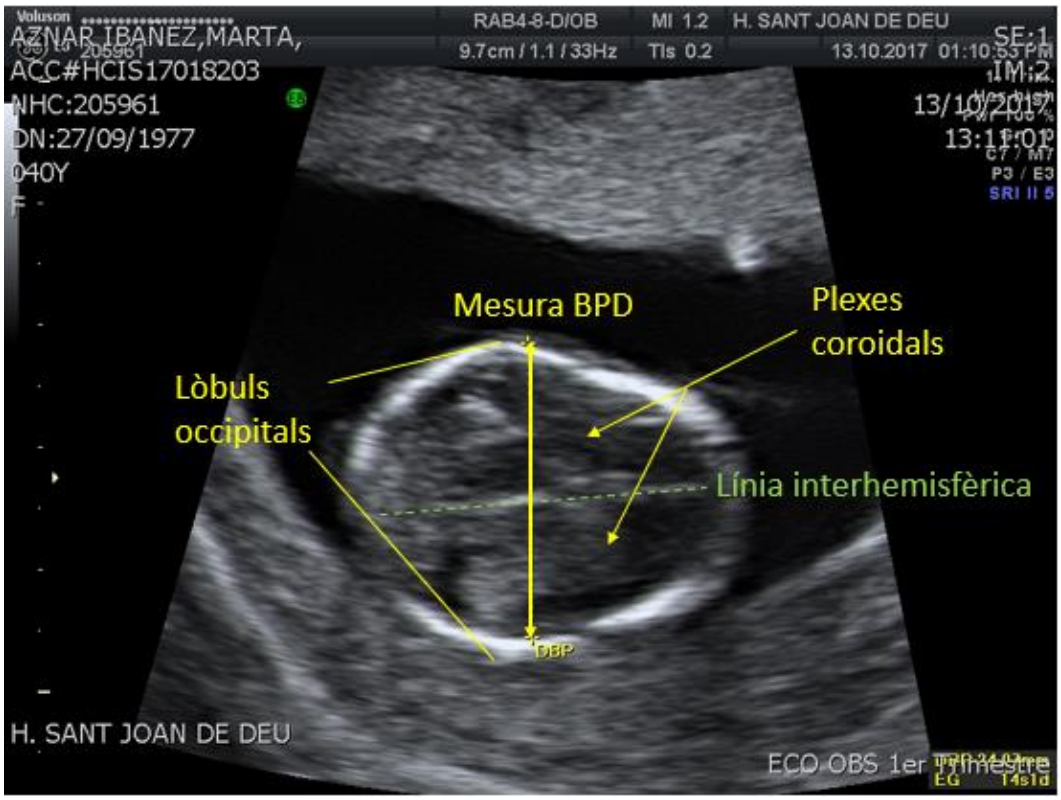


Figura 1-1 Pla Standard del Diàmetre Biparietal (Font: Pròpia)

- **Longitud Crani-caudal – CLR**

En ecografia cerebral el mètode utilitzat per determinar la edat gestacional de l'embrió és la mesura de la distància CLR. La imatge presa es augmentada fins a ocupar la major part de la pantalla, però sempre respectant la condició de que s'hi vegin clarament els punts finals del cap i del cul. Això, es degut a que la mesura CLR s'obté mitjançant l'ajuda de "calipers" situats de tal manera que delimiten els punts definits per la part més alta del cap i la part més prominent del còccix, és a dir, es tracta de mesurar de la longitud més llarga del fetus.
La posició correcta del fetus per a una bona presa de la imatge és quan el fetus s'hi troba en una posició neutral, ni flexionat ni hiperextés. Per assegurar que el fetus no es troba en flexió, el fluid amniòtic deu ser visible entre el mentó i el pit del fetus. Una mostra de correcta adquisició d'aquest pla s'hi pot observar en la Figura 1-2.

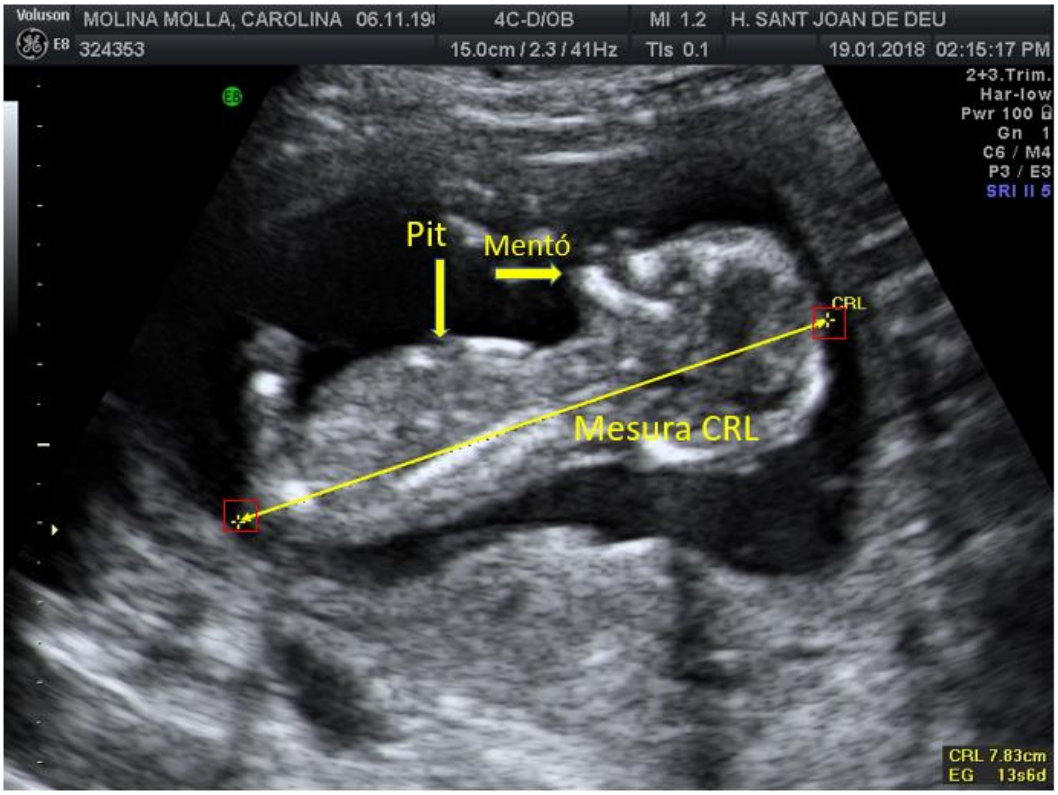


Figura 1-2. Pla Standard de la Longitud Crani-caudal (Font: Pròpia)

Per a estimar l'edat de gestació del fetus, la Societat Catalana d'Obstetrícia i Ginecologia recomana l'ús de la taula publicada per Robinson i Fleming l'any 1975 com a causa dels resultats de les seves investigacions. En casos de gestació normal, el fetus deuria tenir una mesura de entre 42 i 84 mm respectivament. En els casos on la mesura de CLR sigui major a 84mm, les mesures més fiables per establir l'edat gestacional passen a ser les biometries cefàliques com el diàmetre biparietal (BPD).

- **Translucidesa Nucal – NT**

Aquest, és el marcador ecogràfic més sensible per a la detecció de les alteracions cromosòmiques durant el primer trimestre [2]. Això, es degut a que és el marcador amb més pes específic dins el càlcul de risc d'aneuploïdia i per tal de minimitzar les desviacions en la mesura que podrien alterar el resultat del càlcul, cal aplicar una metodologia molt estricta per a la seva correcta mesura, deu ser realitzada per un professional certificat. La Translucidesa Nucal és una acumulació fisiològica de líquid, procedent del sistema limfàtic paracervical, que pot observar-se al clatell del fetus. L'augment del gruix d'aquesta col·lecció líquida es relaciona amb una major prevalença d'aneuploïdies, a més, també pot associar-se amb diverses malformacions fetals, especialment cardiopaties. Una mostra de correcta adquisició s'hi pot observar en la Figura 1-3.

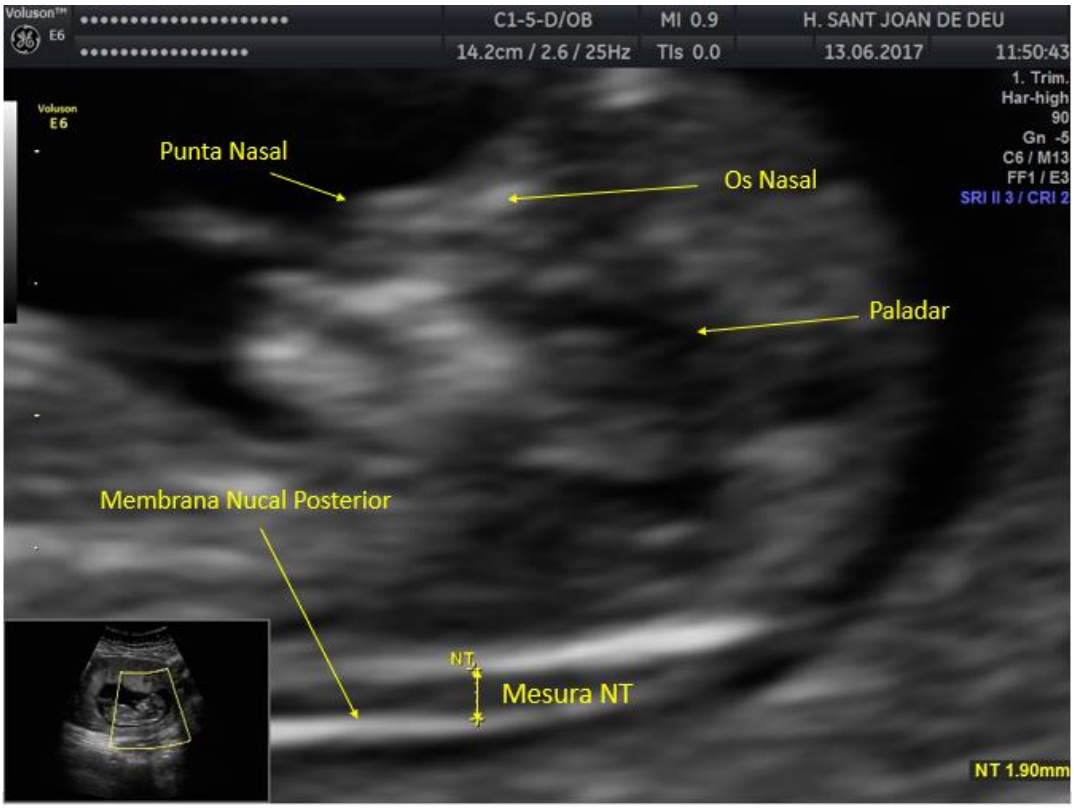


Figura 1-3. Pla Standard de la Translucidesa Nucal (Font: Pròpia)

Per a una correcta mesura de la TN, s'han definit unes pautes per a seguir una metodologia estandarditzada. El fetus deu ser-hi en una posició neutra, amb el cap en línia amb la columna vertebral, ja que quan el coll fetal s'hi troba deflexionat, la mesura pot augmentar falsament mentre que en hiperflexió del coll es pot disminuir falsament. S'ha d'obtenir un bon tall sagital del fetus, definit per la presència de la punta ecogènica del nas i la forma rectangular de la part anterior del paladar, el

diencèfal translúcid al centre i la membrana nucal posterior. I finalment procedir a una ampliació de la imatge per tal que sols ocupin la pantalla el cap i tòrax. En quant a la mesura, s'ha de realitzar en la part més ampla de la NT, mitjançant una correcta col·locació dels "calipers" electrònics entre la vora externa del teixit tou i la vora interna de la pell.

Aquest procediment és estàndard i es fa de forma protocol·litzada per a efectuar un seguiment de l'evolució del fetus. Per a la seva realització es requereix de recursos humans altament qualificats experts en ecografia.

Es planteja per part d'un metge del servei de obstetrícia de l'Hospital Sant Joan de Déu la possibilitat de desenvolupar un sistema automàtic. Aquest sistema permetria que els professionals es dediquessin a tasques de major valor afegit clínic, i a més, proporcionaria una millor robustesa davant variabilitat entre instruments, pacients, equipaments.

1.2. Estat de l'Art del problema

La Visió per Computador, i dins d'ella, la classificació automàtica d'imatges, és una de les principals àrees de treball de la Intel·ligència Artificial. I el Machine Learning proveeix les eines necessàries per dur a terme aquest tipus de tasques. El principi fonamental del funcionament d'aquests mètodes radica en la necessitat d'un conjunt de dades per construir un model amb la capacitat de realitzar tasques automàticament. De fet, l'eficàcia d'aquests mètodes es troba estretament relacionada amb la mida del conjunt de dades que s'utilitza com a entrenament [3][13] trobant que per quan major sigui aquest, millors seran els resultats que s'obtindran. El ventall de possibilitats que ofereix el Machine Learning és molt divers, des de la classificació automàtica, passant pel reconeixement de patrons i les regressions.

Des dels inicis s'han experimentat un seguit d'innovacions tecnològiques a l'àrea de la classificació automàtica, com per exemple, abans els mètodes es trobaven basats en atributs escollits manualment com estadístics de segon ordre o en l'àmbit de la Visió per Computador, la segmentació manual de les regions d'interès, o en alguns casos, una mescla d'ambdues. Amb aquesta selecció d'atributs era possible aconseguir una interpretació precisa dels diferents tipus d'escenaris, sent capaços de discriminar les dades segons el contrast, el color, formes o la textura. Un dels avantatge que presentava aquesta metodologia d'actuació era que es requeria de menys memòria, i per tant, s'acabava disposant d'un model capaç, encara que moltes vegades amb limitacions, de realitzar la tasca desitjada sense un

cost computacional molt excessiu. Degut a la presència de limitacions, que a mesura que es pretenien tractar problemes més complexos anaven en augment, en deteriorament de l'eficàcia, es va canviar

l'enfoc. Varen passar a utilitzar uns models d'aprenentatge automàtic que requerien de tot el conjunt íntegre de dades, provocant un augment considerable de la memòria necessària per a desenvolupar el procés d'aprenentatge, i llavors, com que ja no es realitzava una selecció prèvia de amb quins atributs o patrons es centraria el model, van tenir que aprendre a trobar per si mateixos patrons interns en el conjunt de dades i a distingir quins són els específics de cada classe. Com es pot veure, el cost computacional d'aplicar a un d'aquests models un procés d'aprenentatge és va disparar.

En quant a la utilització d'aquests mètodes en tasques de classificació d'imatges ecogràfiques, s'ha trobat diversos estudis que mostren que a pesar de les propietats no avantatjoses de les imatges obtingudes a partir d'ones acústiques, es possible aconseguir models amb una eficàcia elevada.

Aquests estudis consistien en desenvolupar un model amb la capacitat de reconèixer si un nòdul tiroidal es benigne o maligne. Per a això, es va tenir que entrenar el model amb un gran conjunt d'imatges de nòduls tiroidals benignes, i un altre conjunt de malignes. Aquest, a diferència del nostre es tracta d'un problema de classificació binària, però que ja ens dona una idea de la viabilitat de l'ús d'aquest tipus de tecnologia. Els resultats dels dos estudis són els següents, per al primer varen utilitzar una xarxa neuronal amb una sensibilitat del 92.31%, una especificitat del 76% i una eficàcia de 84.74% [7].

Al segon estudi, es va utilitzar una xarxa neuronal de convolució creada reaprofitant part d'una de ja entrenada per a un altre tipus de problema, amb la qual es va obtenir una eficàcia en la classificació de 98.3%, una sensibilitat del 99.1% i una especificitat del 93.9% [8]

Tot i això, també s'ha analitzat tot el treball i resultats obtinguts per la Safae Bendali al seu Projecte de Fi de Grau [19], on es tractava el mateix problema, però on en compte de l'ús de Xarxes Neuronals de Convolució que s'han utilitzat en el present treball, utilitzava una altra tècnica de Machine Learning anomenada Bag of Words, l'Anàlisi de Components Principals (PCA) i K-Nearest Neighbours. Utilitzant aquestes tècniques es va aconseguir un 83.7% d'efectivitat en la classificació de les imatges corresponents als tres plans ecogràfics tinguts en compte. Com que ja disposàvem dels resultats d'aplicar tots aquests mètodes per a la resolució d'aquest problema, es va decidir que per al desenvolupament d'aquest projecte s'utilitzarien directament uns altres tipus de mètodes.

L'estudi d'aquests experiments mostra les grans possibilitats que ofereix el Machine Learning a l'hora de realitzar tasques com aquesta. Com s'ha vist, la Safae al seu TFG va aconseguir uns molt bons resultats tenint en compte que es tracta d'un problema de classificació multivariable, que sempre resulta més complex que un de binari, i que no disposava d'un conjunt d'imatges molt extens. Per tant, en aquest projecte, el que es pretén realitzar és trobar una altra solució més eficaç per a aquest problema, però aquesta vegada amb la utilització de Xarxes Neuronals de Convolució, que com hem vist, el seu ús pot suposar una certa millora en els resultats.

2. Objectius del treball

2.1. Especificacions i Requeriments

L'objectiu d'aquest treball és trobar un mètode amb la capacitat de realitzar una classificació eficaç dels diferents Plans Standard Biomètrics Fetals que s'han tingut en compte, BPD (Diàmetre Biparietal), CRL (Longitud Crani-caudal) i NT (Translucidesa Nucal). A continuació es pot observar un breu resum de les tasques necessàries per a aconseguir-ho:

- **Obtenció del Conjunt d'Imatges**: Adquisició, etiquetatge i preprocessat d'un conjunt suficientment gran d'Imatges.

- **Elecció i confecció dels models de classificació:** Selecció i condicionament dels models de Machine Learning elegits per a fer-los apropiats per a treballar amb les imatges del nostre Conjunt de Dades. Aquests, que tenen en comú la utilització de Xarxes Neuronals de Convolució (CNN), es divideixen respectivament en dos mètodes segons l'ús que se li doni a les xarxes. Un dels mètodes consisteix en la utilització d'una CNN com a extractor de característiques de les imatges, seguit d'una classificació amb un SVM. El segon dels mètodes consisteix en la utilització d'una CNN per a realitzar totes les parts del procés.

- **Entrenament i classificació de Plans Biomètrics Fetals**: utilitzant diverses CNN s'han seguit els fluxos de treball dels dos mètodes per realitzar un estudi de l'eficàcia en la tasca de classificació de les imatges del Conjunt. Mitjançant aquest estudi es trobarà quina és la millor solució per a la resolució del problema.

2.2. Entorn de prorgamació utilitzat

Per al desenvolupament d'aquest projecte s'han utilitzat Python i MATLAB com a llenguatges de programació. Ambos proporcionen una molt bona interfície de treball i un gran conjunt d'eines i prestacions a l'hora de realitzar tasques relacionades amb la Intel·ligència Artificial, ja que disposen de diferents llibreries dissenyades per a treballar amb Visió per Computador, Machine Learning, Processament d'Imatges i Estadística entre altres, a més d'un entorn de

desenvolupament informàtic numèric integrat. Encara que segons per a que, l'ús d'un llenguatge o l'altre pot presentar més avantatges, això es degut a que Python, com a software lliure, facilita molt l'accés a tot tipus de programes desenvolupats tant per professionals com per aficionats, aspecte que augmenta considerablement la versatilitat de les eines disponibles. En canvi, MATLAB al no ser software lliure, sols pots utilitzar allò que estigui implementat en les seves llibreries, en aquest sentit té una versatilitat més restringida, però per contra l'ús d'aquestes eines es troba més optimitzar per a facilitar la feina.

3. Metodologia

3.1. Conjunt d'Imatges Ecogràfiques

El conjunt d'imatges utilitzat durant l'elaboració del projecte han sigut proveïdes quasi en la seva totalitat per l'Hospital Sant Joan de Déu de Barcelona a través d'una col·laboració establerta entre el LASSIE LAB i el departament d'obstetrícia. Aquestes imatges han sigut preses mitjançant tres diferents equips d'ultrasons, GE Voluson-E6 [5], GE Voluson-E8 [6] i Siemens Acuson-S2000 [18]. A aquestes imatges hem afegit aquelles que varen ser utilitzades per a l'elaboració del projecte del que aquest és continuació. Les quals van ser també proveïdes en part per el departament d'obstetrícia, i unes quantes que va aconseguir a través de fonts públiques online d'imatges amb la intenció d'augmentar la precisió de la classificació, objectiu que també ha motivat la decisió d'incloure-les en la elaboració del present projecte. Ja que degut a les característiques i limitacions del Machine Learning, un increment significatiu del conjunt de dades comporta una millora de la tasca de classificació, és a dir, el model és capaç de generalitzar i així respondre acuradament front a noves imatges no vistes.

Les imatges varen ser proveïdes dins de carpetes, on cadascuna corresponia a una sessió d'ecografia, per tant va ser necessària una tasca de selecció i etiquetatge d'acord amb les respectives categories tingudes en compte en el present projecte: BPD, CLR i NT; descartant totes aquelles que no eren representatives de cap dels anteriors grups.

Classes del Conjunt d'Imatges	Equipament			Imatges Totals
	GE-E6	GE-E8	Siemens	
BPD	155	113	138	406
CRL	160	109	146	415
NT	163	117	127	407

Taula 3-1. Distribució de l'origen del Conjunt de Dades de nova adquisició (Font: Pròpia)

Classes del Conjunt d'Imatges	Equipament			Imatges Totals
	GE-E6	Siemens	Online	
BPD	27	25	48	100
CRL	28	24	46	98
NT	28	24	49	101

Taula 1-2. Distribució de l'origen del Conjunt de Dades inicial (Font: Pròpia)

Classes del Conjunt d'Imatges	Origen		Imatges Totals
	Vicent	Safae	
BPD	406	100	506
CRL	415	98	513
NT	407	101	508

Taula 1-3. Distribució de l'origen del Conjunt de Dades (Font: Pròpia)

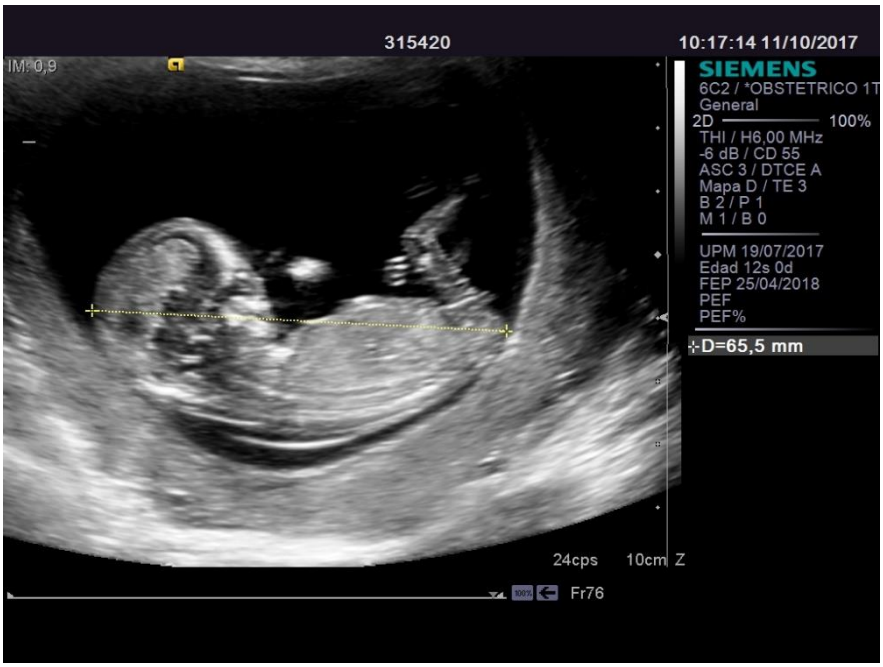
Com es pot observar en les taules presentades, s'ha aconseguit incrementar el nombre total d'imatges, punt que es considerava molt important al començar a desenvolupar el projecte, ja que és un dels pilars clau per a un bon funcionament del Machine Learning. Al començament es disposava de les quantitats que veiem en la Taula 1-2, però a aquestes vàrem afegir les de la Taula 3-1.

Un dels requisits del Machine Learning és que el conjunt de dades s'hi trobi balancejat, és a dir, que front a un problema de classificació amb més de dos classes, és necessari treballar amb el mateix nombre de mostres per a cadascuna de les classes, obtenint així uns resultats que generalitzen molt millor. Per tant, com que la classe de la que menys mostres hi tenim és la BPD, amb 506 classes, en el moment de fer córrer el model, hi utilitzarem sols 506 mostres de cada classe, descartant les sobrants dels grups CLR i NT.

En aquesta secció, seran presentats exemples de les Imatges d'Ecografies que componen el conjunt de dades final. Com s'ha dit anteriorment, de totes les imatges de les diferents sessions de que disposàvem és varen seleccionar aquelles que representaven els diferents plans desitjats per al problema en qüestió, fent un etiquetatge exaustiu. En la Figura 3-1 es presenta un exemple del tipus BPD. On es poden observar clarament les diferents estructures, com el cap fetal, el qual es troba bastant centrat en aquesta imatge (casuística no molt comuna, degut al gran espectre d'imatges obtingudes per mitja de diferents equips i amb diferents professionals), i l'àrea negra hipoecoica que representa el fluid intrauterí, la qual es troba rodejada per una altra àrea de diferents característiques, amb una tonalitat més brillant i amb una textura no tan homogènia, que representa les diferents parts de l'úter.

Figura 3-1. Mostra de les Imatges del tipus BPD (Font: Pròpia)

En canvi, la Figura 3-2 presenta un exemple de la classe CRL, com podem observar, ara es representa un pla totalment diferent al anterior, on l'important era proporcionar una imatge per calcular la mida del Diàmetre Biparietal, i ara en canvi es busca trobar el fetus de cos sencer i en una postura neutral per poder calcular la mida CRL.



Faula 3-2. Mostra de les Imatges del tipus CRL (Font: Pròpia)

Per últim, en la Figura 3-3 trobem un exemple referent a la classe NT, que ens mostra un altre pla ecogràfic bastant similar al anterior, però que en compte d'una vista del cos sencer es centra en la meitat superior del fetus, tenint el plec nucal com a referència, ja que és la mesura que es pretén trobar.

Per a la realització de dels experiments s'ha fet una partició del conjunt de dades, deixant un 80% per al entrenament dels models (406 mostres de cada classe) i un 20% per a la comprovació del funcionament del model (100 mostres de cada classe). Aquesta partició serà la que s'utilitzarà generalment al llarg de tots els experiments tinguts en compte, menys en el cas que compren l'elaboració d'una Xarxa Neuronal de Convolució, on es va tindre que realitzar una tercera partició de les dades per a que al modificar l'estructura de la xarxa amb la finalitat de millorar els resultats, per assegurar que la millora dels resultats és real i que al canviar l'estructura això no porta associat un aprenentatge no genèric. En el cas que s'acaba d'exposar la partició va ser la següent per cada classe: 386 mostres per a l'entrenament, 20 mostres per a la testejar l'evolució del model al canviar l'estructura i 100 mostres per a la comprovació final.

Figura 3-3. Mostra de les Imatges del tipus NT (Font: Pròpia)

3.2. Entenent el Machine Learning

3.2.1. Machine Learning

El concepte d'Aprenentatge Automàtic (Machine Learning) s'utilitza per definir el camp de la Intel·ligència Artificial i de les Ciències de la Computació que s'encarrega de desenvolupar tècniques per a ensenyar als ordinadors a realitzar tasques naturals per a les persones. És a dir, es tracta d'aconseguir programes capaços de generalitzar comportaments a partir d'un pas previ on aprèn a fer-ho per mitjà d'exemples.

Concretament, la tasca que portarà a terme en aquest projecte es tracta d'una tasca de classificació automàtica per mitjà de l'aprenentatge supervisat, que consisteix en que el model aprengui les relacions entre les entrades d'entrenament i les seves etiquetes.

L'aprenentatge automàtic suposa un gran canvi respecte a les tècniques utilitzades a la programació clàssica, on el programador havia de proveir les regles de funcionament del programa i la informació, i amb això el programa hi proporcionava una solució. En canvi, el pas al Machine Learning va suposar un gran avanç, ja que aquests sistemes són entrenats en comptes de ser programats, Figura 3-4. Els models troben estructures estadístiques a partir de l'anàlisi exhaustiu de molts exemples, la qual cosa, permet al sistema trobar una sèrie de regles per a automatitzar la tasca, és a dir, dit d'una altra manera, per executar una tasca de processament d'informació. Per tant, per a treballar amb aquest tipus de tecnologia, el que es necessita és:

- Dades d'entrada
- Exemples de les sortides esperades
- Una forma de mesurar si realment l'algoritme està fent una bona tasca d'automatització. Utilització d'una senyal de realimentació → Aprenentatge

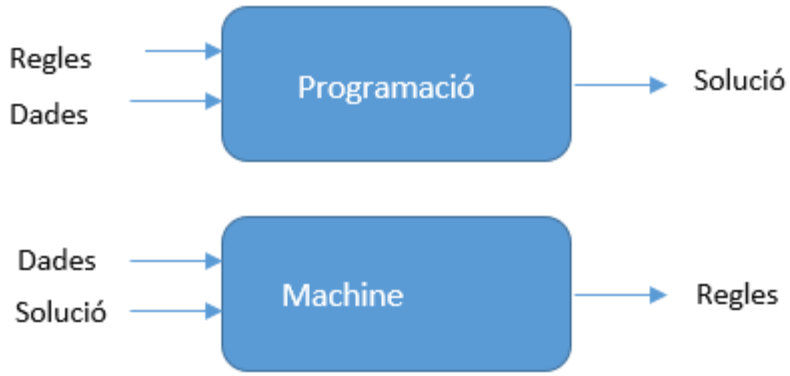


Figura 3-4. Comparació amb Workflow tradicional (Font: Pròpia)

El principal problema que presenta l'Aprenentatge Automàtic és que el model aprengui representacions útils de les dades d'entrada per a poder generalitzar millor i així obtindré unes sortides (solucions) correctes. Aquestes representacions consisteixen en transformacions aplicades a les dades que les fan més disposades per a una tasca determinada, que en el cas que tractarem serà una tasca de classificació. Per tant, els algoritmes utilitzats tracten de trobar aquelles transformacions que transformen les dades a unes representacions més útils per a desenvolupar el treball.

Concretament, s'han utilitzat tècniques d'aprenentatge profund (Deep Learning), que consisteix en una estructura matemàtica per a permetre l'aprenentatge de representacions de dades. Això, ho fa utilitzant arquitectures compostes de transformacions no lineals múltiples. Aquí, l'arquitectura que s'utilitzarà per a la tasca de classificació seran les Xarxes Neuronals de Convolució, arquitectura que s'explicarà en els següents apartats.

3.2.2. Xarxes Neuronals

Les xarxes neuronals construeixen un mapa per enllaçar les dades d'entrada amb una de les possibles sortides, que en el cas dels problemes de classificació, és una de les etiquetes especificades. Això, ho aconsegueix a través d'una seqüència de transformacions simples de les dades d'entrada, aquestes transformacions és allò que el model aprèn per la exposició als exemples.

A la Figura 3-5 hi podem veure la estructura general d'una xarxa neuronal, on les neurones són la unitat estructural bàsica.

Els layers (capes formades per neurones) són la estructura (el cos) del model, contenen les dades fonamental que defineixen les xarxes neuronals. La construcció de models d'aprenentatge profund és basa en l'adició de layers compatibles, per tal de formar canals de transformació de les dades. Cadascuna d'aquestes capes sols accepta l'entrada de tensors d'una certa mida i en retorna d'altres amb una mida diferent, per això és necessari que els layers que s'hi van afegint al model tinguin unes dimensions concretes.

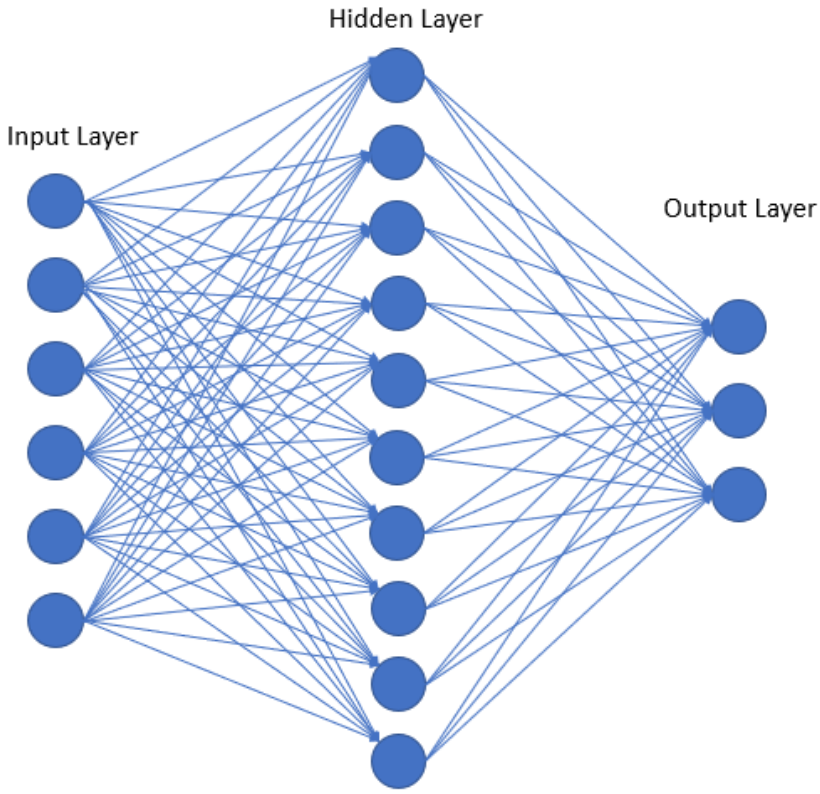


Figura 3-5. Estructura general d'una xarxa neuronal (Font: Pròpia)

En el cas de la Figura 3-5, s'hi observa un model format per tres capes. La primera (Input Layer) consisteix en la capa que contindrà les dades de la mostra d'entrada (tant pot ser una sèrie temporal, com una imatge d'una, dos o tres dimensions o també una sèrie de propietats), la segona (Hidden Layer) correspon a la capa que aplica les transformacions que contenen les neurones a les dades d'entrada. Aquestes transformacions venen especificades pels pesos del layers (weights, determinen que li fa un layer a les dades d'entrada, també conegut com a paràmetres del layer). Per últim, s'hi troba el Output Layer, que per al cas que analitzarem actuarà com a classificador, aquest, segons el

resultat obtingut després de les transformacions aplicades pel Hidden Layer a les dades d'entrada classificarà cada mostra en la classe que hagi après que s'assembla més.

L'aprenentatge, consisteix llavors en trobar el conjunt de valors dels pesos de les diferents neurones de la xarxa per a que aquesta produeixi un mapa acurat de les entrades amb les seves corresponents etiquetes.

A la Figura 3-6 hi podem veure la estructura general del funcionament de les xarxes neuronals. S'hi pot observar l'estructura tractada abans, on la mostra de dades passa per un seguit de layers, que segons el resultat que s'obté després d'aplicar les transformacions corresponents als pesos d'aquests, el model fa una predicció de la classe a la que pertany la mostra. Aquesta sortida que obtenim es compara amb la etiqueta correcta de la mostra, la comparació es fa a través de la funció de pèrdues, que no és més que una funció objectiu que mesura com de lluny es troba la sortida que s'ha obtingut del que realment s'esperava. Així que aquesta funció agafa la predicció i la etiqueta correcta i computa el valor de distància entre ambdues.

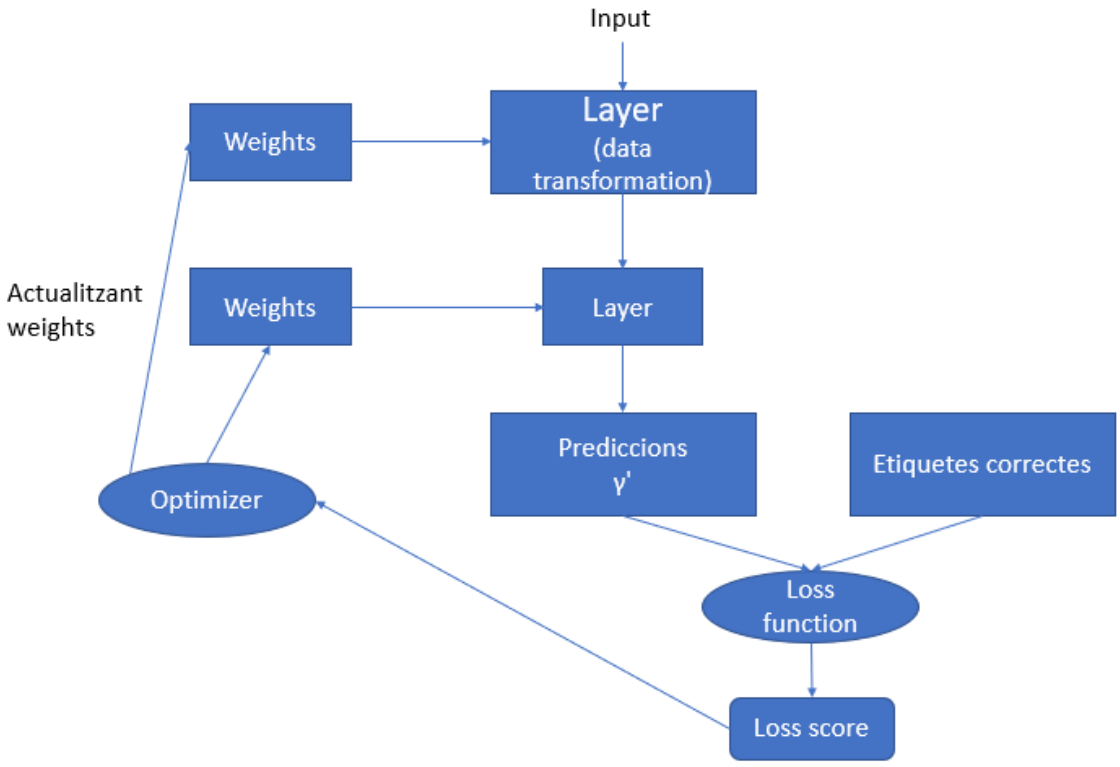


Figura 3-6. Esquema del procés d'aprenentatge d'una xarxa neuronal (Font: Pròpia)

Aquest valor que s'obté de la funció objectiu és el que s'utilitza per ajustar el valor dels pesos. Ajustament realitzat a través del optimitzador, que implementa un algoritme de Backpropagation (algoritme de optimització), que determina com s'han d'actualitzar els pesos de la xarxa. Llavors, hi entren més mostres al model, i es torna a realitzar tot el procés explicat fins assolir un valor determinat

d'exactitud en la classificació, un de la funció objectiu o fins un determinat nombre de iteracions. Per tant, el procés d'optimització es correspon a una optimització basada en gradients.

3.2.3. Optimització per gradients

Un exemple de les possibles transformacions que s'hi poden aplicar a les dades d'entrada seria la representada a l'Equació 3.1:

$$\boldsymbol{output = relu(dot(W, input) + b)} \quad \textbf{(Eq. 3.1)}$$

On W i b són els paràmetres entrenables dels layers, pesos i biaix corresponentment. Aquests paràmetres contenen la informació apresa per la xarxa a base de la seva exposició a les dades d'entrenament.

A l'inici del procés d'aprenentatge d'una xarxa neuronal, aquests paràmetres s'hi troben inicialitzats aleatòriament generalment amb valors petits. El que s'entén com a procés d'aprenentatge és el posterior ajustament d'aquests pesos a través d'una senyal de retroalimentació que obtenim durant l'entrenament. Aquests passos ocorren a dins d'un llaç d'entrenament que va repetint els següents passos tantes vegades com sigui necessari:

1- Agafar un lot de mostres d'entrenament 'x' i els seves corresponents etiquetes 'y'.
2- Fer córrer la xarxa en 'x' i obtenir prediccions 'y_pred'.
3- Calcular el valor de pèrdua de la xarxa a través de la funció objectiu per al lot de mostres seleccionat i mesurar la discordança entre 'y' i 'y_pred'.
4- Actualitzar tots els pesos de la xarxa d'una manera que redueixi un poc les pèrdues en aquest lot de mostres.

Un dels punts més clau en Deep Learning és trobar una bona actualització dels pesos, per tant, com es pot calcular si el coeficient deu ser augmentat o disminuït i quant?

Això, s'aconsegueix gràcies a que totes les operacions utilitzades a la xarxa són diferenciables, així que calculant el gradient del valor de la funció objectiu respecte als pesos de la xarxa en aqueix moment i actualitzant aquests coeficients en la direcció oposada a la marcada pel gradient, s'obté una disminució de la pèrdua.

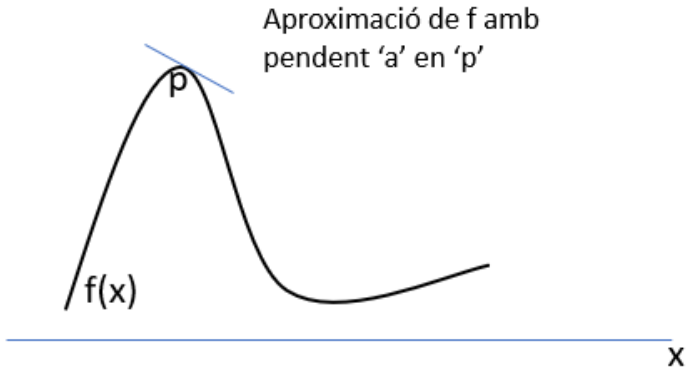


Figura 3-7. Representació d'una funció d'una variable (Font: Pròpia)

A la Figura 3-7 s'hi representa la idea general del que es pretén realitzar, prenem com a exemple aquest cas per entendre-ho. L'objectiu és minimitzar el valor resultant de la funció objectiu 'f(x)', que varia segons els diferents valors del pes 'x', això s'aconsegueix calculant el gradient de la funció en el punt p (punt de partida) i actualitzant així el valor 'x' canviant el seu valor cap al costat contrari d'on marqui el gradient, fins arribar al mínim on al calcular el gradient, s'hi troba que per a moure's cap als dos costats s'ha de pujar pendent → s'ha assolit el mínim.

Aquesta és la síntesi del funcionament del mecanisme d'optimització dels pesos, on realment el que ens trobem no és una sola variable (pes), sinó que presenta un gran conjunt de pesos, on per a cada capa de la xarxa s'hi troba una estructura de pesos tal que així: W[i, j]. On 'W' correspon a la matriu que conté tots els pesos de la capa en qüestió. Per tant, realment, el que s'ha d'optimitzar és el resultat d'una funció objectiu en un espai multidimensional com la representada a l'Equació 3.2.

$$\boldsymbol{f(w1, w2, w3, \dots, wn) \rightarrow funció\ a\ optimitzar} \qquad \textbf{(Eq. 3.2)}$$

Una funció d'aquestes característiques s'ha de derivar utilitzant la regla de la cadena, l'aplicació de la qual per al càlcul dels valors de gradient d'una xarxa neuronal dona lloc a l'algoritme de Backpropagation. Una vegada calculat el valor de pèrdua de la funció objectiu, comença calculant el gradient cap enrere des de les capes superiors fins a les inferiors, aplicant la regla de la cadena per trobar la contribució que té cada paràmetre en la funció.

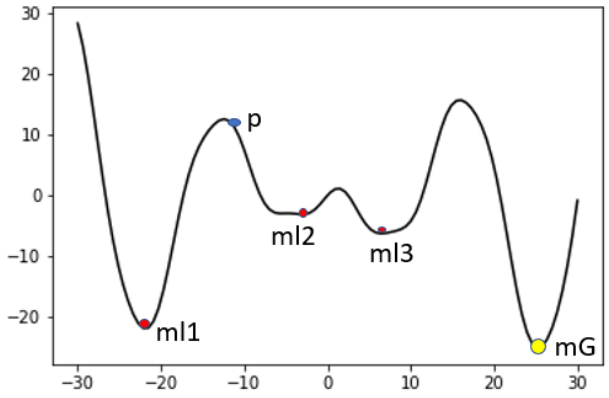


Figura 3-8. Representació visual del problema dels mínims locals(Font: Pròpia)

Trobar la combinació de pesos que proporciona el mínim valor possible de la funció objectiu pot no ser una tasca tan senzilla com en el cas de la Figura 3-7, on sols hi ha un mínim, ja que la funció pot ser com la que es presenta a la Figura 3-8. On en el suposat cas de trobar-se en el punt 'p', amb la tècnica que s'ha explicat s'arribaria al mínim local 2, que no és ni el més petit entre els locals, deixant així, el mínim global sense explorar, és a dir, agafant el 'ml2' com a la millor solució. Per tant, el que s'utilitza per tal d'evitar l'estancament en un mínim local és un algoritme d'optimització més complert que el de Backpropagation, com el de Descens per Gradient Estocàstic, que s'explicarà a continuació. Aquest, tot i ser l'algoritme més comú per a Deep Learning hi ha d'altres més potents per a segons quin tipus de problema.

3.2.3.1 Gradient de descens estocàstic

És una variant del explicat a l'apartat anterior, que és molt útil per aproximacions en aquells casos que els pesos ja hi són molt pròxims als que constitueixen el mínim. La novetat que aquest cas incorpora és el factor de salt (step), que permet una convergència molt més gran per a casos on el mínim global de la funció s'hi troba molt lluny.

Llavors, per reduir f(W[i,j]), s'actualitzaran els pesos en la direcció contrària a la trobada al calcular el gradient, com indica l'Equació 3.3:

$$\boldsymbol{W1 = W0 - step * gradient(f(W0[i,j]))} \qquad \textbf{(Eq. 3.3)}$$

Aquí, és molt important realitzar una selecció molt raonable de la magnitud del salt (step factor), ja que agafar un valor molt petit implicaria tenir que realitzar massa iteracions fins a convergir, i que una vegada s'haja convergit, sigui a un mínim local. Per contra, agafant un valor molt gran implicaria el salt a posicions completament aleatòries en la funció.

3.2.4. Visió General

Com s'ha anat explicant al llarg dels anteriors apartats, les claus per configurar el procés d'aprenentatge són:

→ La funció de pèrdues (funció objectiu), representa una mesura de l'èxit de la tasca duta a terme. Proporciona la funció a minimitzar durant el procés d'aprenentatge.

Segons el tipus de problema, la funció objectiu que s'elegirà serà una o un altra. Aquí hi ha uns quants exemples:

- Binary crossentropy: per a problemes de classificació binària.

- Categorical crossentropy: per a problemes de classificació amb més de dos classes.

- Mean squared error: per a problemes de regressió.

- Connectionist temporal classification (CTC): per a problemes d'aprenentatge de seqüències.

Segons el problema que s'ha tingut en compte per aquest projecte, la funció objectiu a utilitzar serà la de 'Categorical crossentropy', ja que es tracta de classificar imatges corresponents a tres grups diferents. El terme 'Crossentropy' correspon al camp de les dades, i mesura les distàncies entre distribucions de probabilitats. Per tant, s'ha de minimitzar la distància entre aquestes dos distribucions entrenant la xarxa per a produir sortides tan properes com sigui possible a les etiquetes correctes de les mostres.

→L'optimitzador, determina com s'actualitzarà la xarxa en base a la funció objectiu. Com ja s'ha explicat abans, l'optimitzador que s'utilitzarà serà el Descens per Gradients Estocàstics (SGD).

3.2.5. CNN

Per a problemes de classificació d'imatges, Visió per Computador, el tipus de xarxes neuronals utilitzades són les Xarxes Neuronals de Convolució (CNN), això es degut a les característiques pròpies d'aquests tipus d'estructures i la funcionalitat que proporcionen.

En aquest apartat, s'explicarà el mode de funcionament i les diferències entre els diferents tipus de layers presents en les diferents xarxes que s'han utilitzat durant l'elaboració del projecte.

- Densely Connected layer: aquest tipus de capa és el més utilitzat en les xarxes neuronals simples, encara que també són utilitzats en les parts finals de les CNN. L'operació duta a terme en aquestes és la d'una operació lineal on cada entrada s'hi troba connectada amb totes les

sortides per un pes. Generalment, aquesta operació va seguida per una funció d'activació no lineal com la de l'Equació 3.4. A l'Equació 3.5 trobem el conjunt d'operacions realitzades.

$$\boldsymbol{output} = \boldsymbol{f}.\boldsymbol{activ}(\boldsymbol{dot}(\boldsymbol{W}, \boldsymbol{input}) + \boldsymbol{b}) \quad \textbf{(Eq. 3.4)}$$

$$\boldsymbol{resultat}[\boldsymbol{i},\boldsymbol{j}] \mathrel{+}= \boldsymbol{W}[\boldsymbol{i},:] * \boldsymbol{input}[:,\boldsymbol{j}] \;\rightarrow \boldsymbol{output}[\boldsymbol{i},\boldsymbol{j}] = \boldsymbol{f}.\boldsymbol{activ}(\boldsymbol{resultat}[\boldsymbol{i},\boldsymbol{j}] + \boldsymbol{b}) \quad \textbf{(Eq. 3.5)}$$

Sense la funció d'activació, les capes completament connectades sols consistirien en dos operacions lineals, un producte escalar i una addició. Amb la qual cosa, el model sols aprendria transformacions lineals de les dades d'entrada, i llavors, l'hipotètic espai del layer seria el conjunt de totes les possibles transformacions de les dades d'entrada en un espai de 'n' dimensions. Per tant, amb la intenció d'aconseguir un espai hipotètic molt més ric es necessiten unes representacions més profundes, no lineals. Les dues que s'han utilitzat són:

- La activació 'Relu' (rectified linear unit), funció centrada en zero que fa fora els valors negatius.
- La activació 'Softmax', aquesta funció és la utilitzada en tots els últims layers de les CNN utilitzades per a classificació multiclasse, ja que comprimeix els valors arbitraris que rep a dins de l'interval [0, 1], obtenint com a sortida uns valors que poden ser interpretats com una probabilitat. Llavors, el layer que contingui aquesta activació ens dirà la probabilitat que té la imatge analitzada de pertànyer a cadascun dels grups.

- Layers Convolucionals (Covnets): són els característics de les CNN. La principal diferència que presenten amb els 'Densely Connected'' és que aquests aprenen patrons globals, mentre que els convolucionals aprenen patrons locals. És a dir, els patrons que aprenen no són sensibles a canvis en la ubicació. Per exemple després d'aprendre un patró que es troba en el cantó superior dret de la imatge, les xarxes amb layers convolucionals poden reconèixer aquest mateix patró estigui on estigui. En canvi, els 'Densely Connected' tindrien que tornar aprendre'l altra vegada per la nova ubicació.

La unitat bàsica que permet les operacions a les Covnets són els filtres. Al moment de definir una capa de convolució, se li especifica el nombre de filtres que s'hi vol aplicar a la imatge en aqueixa etapa i les seves dimensions. Per exemple, al definir una capa de convolució com aquesta:

```
layers.Conv2D(32, (3,3))
```

S'hi disposarà d'una capa que aplica la convolució de 32 filtres d'unes dimensions de 3 píxels *3 píxels al llarg de tota la imatge. Per tant, l'entrada d'una d'aquestes capes és la matriu que correspon a la imatge, i com a sortida, s'obtindran 32 matrius d'unes dimensions iguals a les de la imatge original (si s'utilitza 'Padding', que consisteix en afegir un nombre apropiat de files i columnes de zeros a cada costat de la matriu per a fer encaixar la finestra de convolució per a totes les dades d'entrada) o amb unes dimensions files-2*columnes-2, que representen el valor obtingut després d'aplicar la convolució als diferents píxels que constitueixen la imatge.

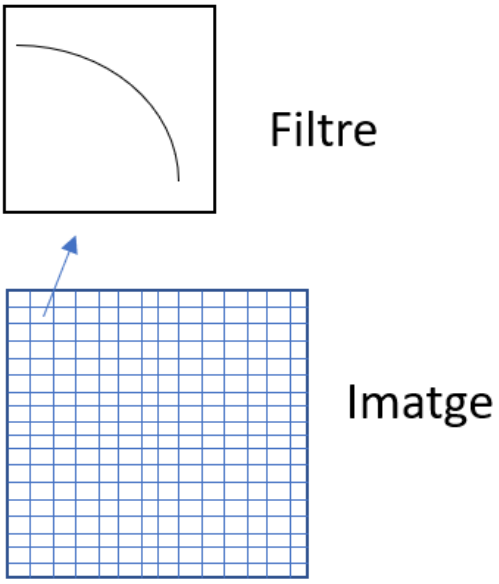


Figura 3-9. Esquema funcionament dels filtres de convolució (Font: Pròpia)

Per exemple, al passar una imatge d'unes dimensions 26*26*1 (en escala de gris), per la capa que hem definit abans, la sortida que s'obtindria seria: 24*24*32 per al cas que no s'ha aplicat el 'Padding', i 26*26*32 per aquells on si que s'ha aplicat. On cadascuna de les ubicacions espacials del mapa de característiques de sortida correspon a la mateixa ubicació del d'entrada.

Una altra de les seves característiques és que poden aprendre jerarquies espacials dels diferents patrons. Una primera capa de convolució aprendrà petits patrons locals com contorns, llavors, una segona capa aprendrà patrons més grans i complexos fets a partir de les característiques extretes a partir de la primera capa, aprenent així uns patrons més abstractes i complexos en cada pas. En la Figura 3-10 hi trobem una explicació més visual de com va augmentant la complexitat de les formes captades en cada capa de la xarxa.

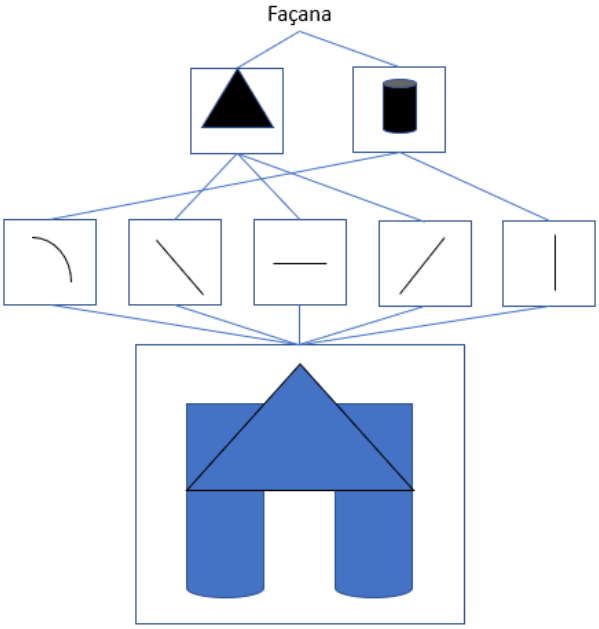


Figura 3-10. Mostra de les Imatges del tipus NT (Font: Pròpia)

Com ja s'ha vist, els paràmetres que s'hi poden establir en un d'aquests tipus de layers són, el nombre de filtres a utilitzar i la dimensió de les finestres on s'aplicarà la convolució, però a més d'aquests, també és possible aplicar una funció d'activació per aquests casos, com és el cas de la 'Relu' explicada a l'apartat anterior. La qual, deixaria a zero tots aquells valors negatius. Un altre paràmetre que pot ser molt útil si es fa servir correctament depenent de les característiques de la imatge és la possibilitat d'aplicar les finestres de convolució en comptes de a cada píxel, fer que deixi una distancia de 'n' píxels entre cadascun al que se li aplica l'operació (convolution strides). La qual cosa resulta en que les dimensions de la matriu de sortida han estat reduïdes.

En el cas d'aquests layers, els pesos que s'aniran actualitzant al llarg de les diverses iteracions del model són les formes dels filtres aplicats per fer les convolucions.

- MaxPooling Layers: la funció que tenen aquestes capes és la de realitzar una reducció dimensional dels mapes de característiques que s'han obtingut a les capes de convolució. És a dir, rep com a entrada els mapes de característiques, dels quals, per a cada canal (mapa resultat de la convolució de cada filtre) aplica una finestra que sol ser de 2*2 amb un factor de salt (stride) de 2, d'on extrau el màxim valor dels presents a la finestra. Amb això, aconsegueix reduir la quantitat d'informació en un factor de 2, quedant-se amb aquelles posicions que per

als diferents filtres han donat uns valors més elevats. Obtenint com a sortida un altre mapa de característiques com el que li havia arribat però amb unes dimensions reduïdes a la meitat i amb la informació corresponent a aquells punts del mapa de característiques que tenien valors màxims respecte als dels seus veïns.

Matriu d'entrada: 26*26*32 → Matriu de sortida: 13*13*32 **(Eq. 3.6)**

- Average Pooling Layers: aquests fan bàsicament el mateix que els 'MaxPooling', però en comptes de quedar-se amb la posició amb el valor màxim, l'operació realitzada als paquets de característiques de cada finestra agafada és la de la mitjana dels valors.

  En general, resulta més útil la utilització del 'MaxPooling Layer', ja que les característiques tendeixen codificar la presència espacial d'alguns patrons al llarg de les diferents peces del mapa de característiques. Per tant, és molt més informatiu mirar la màxima presència de les diferents característiques que la seva mitjana.

- Flatten Layers: aquestes capes serveixen de pont entre les capes convolucionals del principi de la xarxa i les 'Densely Connected' del final. Converteix el format de les dades per a fer-les compatibles a les capes 'Densely Connected'. Per tant, rep els mapes de característiques obtinguts durant les capes de convolució, i va afegint totes les dades ordenadament a dins d'un vector. Així, s'obté un vector per a cada mostra (imatge). Exemple:

  Mapes de característiques: 26*26*32 → Flattened: 1*(26*26*32)=1*21632

- Dropout Layers: aplicat a un layer, la transformació que realitza a les dades que li arriben consisteix en un assignació aleatòria d'un conjunt dels valors de les dades d'entrada als quals se li assigna valor zero durant l'entrenament de la xarxa. El rati del 'Dropout' és la fracció de dades a les que se lis assigna valor zero (normalment entre 0.2-0.5).

  L'objectiu d'aplicar aquesta capa és que la introducció de soroll en els valors de sortida de la capa, pot trencar amb la ocurrència casual de patrons que no són significants per a la tasca en qüestió.

  Pel que fa al moment de comprovar el funcionament de la xarxa (test time), cap de les dades es assignada a zero, en cavi, els valors de sortida de la capa 'Dropout' s'hi troben escalats per un factor reductor per tal d'equilibrar amb l'entrenament, ja que quan al test hi ha més unitats de la capa actives que durant l'entrenament.

- Cross Channel Normalization Layers: aquests tipus de capes apliquen una normalització a nivell local del nivell de brillo al llarg de tota la imatge, cosa que fa que es diferenciï de la normalització de contrast a nivell local, ja que a diferència d'aquesta, no elimina l'activitat mitja.

3.2.6. Metodologies de treball amb CNN

Recordem que l'objectiu del present projecte és el de trobar un mètode eficaç per a la classificació automàtica dels diferents plans d'una sessió d'ecografia fetal, per a això, s'ha fet una comparació dels resultats que dona treballar amb diferents Xarxes Neuronals de Convolució i com varia l'efectivitat de la classificació segons el tipus de xarxa i com es fa servir.

En els apartats anterior hem vist com operen les Xarxes Neuronals de Convolució des d'un punt de vista teòric, per tant, en el present apartat es tractarà de veure quins criteris s'han utilitzat per a l'elecció de les xarxes utilitzades i com s'han fet servir, i els passos que s'han seguit durant l'elaboració de la xarxa que s'ha creat durant.

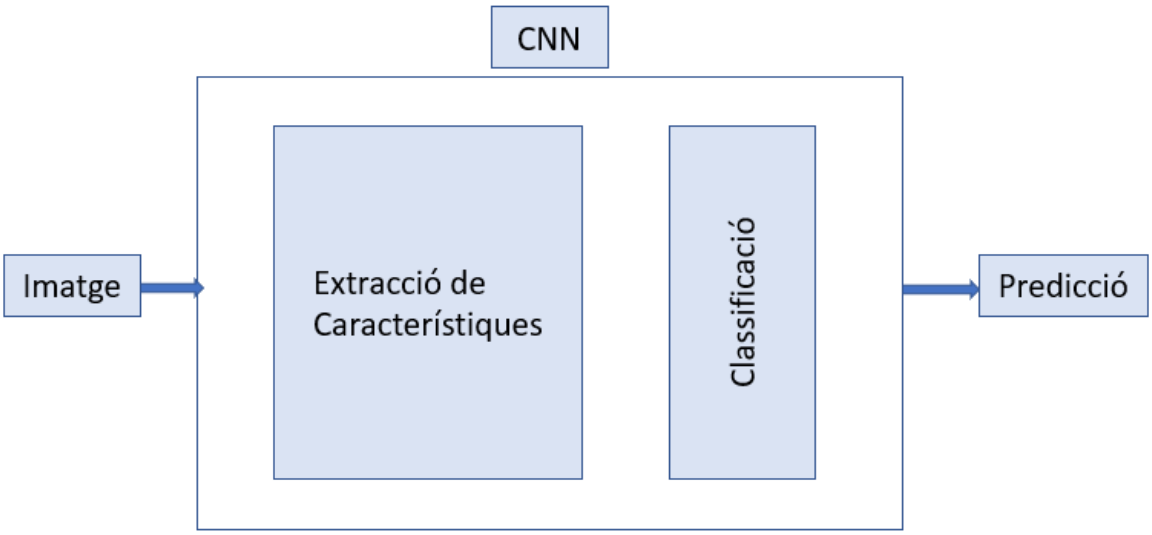


Figura 3-11. Codi Python per definir la versió final de la CNN (Font: Pròpia)

A la Figura 3-11 s'hi observa el mode de funcionament en aquells casos on s'utilitza una CNN per a la realització de tot el procés. Això es degut a que una part de les seves capes funcionen com a Extractors de Característiques, i unes altres com a classificadores.

Per tal de realitzar una tasca de classificació com la que tenim entre mans, la resolució del problema s'hi pot encarar de diferents formes segons la modalitat de CNN que agafem. Això, és degut a que nosaltres mateixos podem construir-nos en una de pròpia des de zero, tenint que perfilar l'estructura i els paràmetres que definiran la xarxa, però també hi ha la opció de utilitzar xarxes de convolució realitzades per altres persones que han resolt problemes pareguts als nostres. Per tant, trobem que hi ha dos opcions:

- Pretrained CNN
- Custom CNN

3.2.6.1 CNN Preentrenades

Com be sabem, Python és un llenguatge de programari lliure, on els avanços que hi van fent els diferents usuaris normalment són compartits, cosa que al final resulta en un benefici comú. Per tant, no és d'estranyar que aquelles Xarxes Convolucionals que han sigut realitzades per grups de recerca o per particulars i que tenen una molt bona efectivitat siguin incorporades a les diferents llibreries de Python (Keras, TensorFlow). Aleshores, una molt bona opció per a quan s'ha de resoldre un problema d'aquest tipus és la utilització una d'aquestes xarxes ja que han sigut entrenades per a la resolució de problemes similars. MATLAB en canvi no és lliure, això comporta per exemple que no totes les xarxes neuronals que s'han anat utilitzant i millorant hi siguin disponibles, sols hi ha unes quantes que han sigut implementades pels desenvolupadors de la plataforma.

Les xarxes utilitzades han sigut: AlexNet (Matlab), InceptionV3 i VGG19 (Python).

El que hi trobem, és que s'hi pot utilitzar tant l'estructura de la xarxa elegida com els seus pesos, resultants del procés d'aprenentatge amb les dades del problema per al que foren creades. Aquesta pràctica permet estalviar molt de temps de computació per part dels ordinadors (que sinó han d'aprendre un nombre de pesos de l'ordre dels milions), ja que la xarxa en comptes de començar a aprendre des de zero, se li pot transferit el coneixement resultat del procés d'aprenentatge anterior. Factor molt important, ja que tant al inicialitzar una xarxa neuronal, tots els pesos de les diferents capes de la xarxa s'inicialitzen amb un valor aleatori, comportant la possibilitat de que els filtres busquen uns patrons sense sentit.

Un altre avantatge de la utilització de xarxes preentrenades és que es tracta del mètode més efectiu per a quan es disposa d'un conjunt de dades no molt gran. Per entendre que s'entén per conjunts de base grans i petits, utilitzarem la comparació entre el nostre conjunt de dades i l'utilitzat per entrenar les xarxes preentranades que s'han utilitzat.

	Nombre classes	Mostres per classe	Total
Vicent	3	506	1518
ImageNet	1000	1200	1200000

Taula 3-4. Comparació rati Imatges/NumClasses (Font: Pròpia)

Les xarxes preentrenades s'han entrenat amb la base de dades 'ImageNet', la qual consisteix en unes 1000 classes diferents on cadascuna té 1200 imatges [1] , que van des d'animals fins a objectes quotidians. El nombre de les nostres mostres al costat de la suma total de les d'ImageNet constitueix un nombre irrisori.

Que s'hagin entrenat amb formes i contorns que no corresponguin als d'imatges mèdiques no és un factor crític, ja que dins de les imatges de la nostra base de dades també hi són presents formes que compleixen amb els patrons que capten les diferents capes de convolució. Al final, allò important d'utilitzar-ne una ja entrenada és aprofitar els pesos que constitueixen els filtres apresos per a les convolucions, així, la xarxa actua com un model genèric de visió del món real.

Hi ha dues maneres diferents d'utilitzar les CNN preentrenades, utilitzar-les com a Extractor de Característiques (Feature Extraction) o reentrenar-les per afinar-les amb el nou problema (Fine Tuning).

- Feature Extraction: bàsicament consisteix en aprofitar les representacions apreses per les xarxes elegides amb la finalitat d'utilitzar-les com a extractores de característiques interesants de noves mostres.

  El que es fa és prendre tot el bloc de convolució de la xarxa escollida, deixant de banda l'última part de la xarxa, que és la que actuava com a classificador. Per tant, al fer córrer les noves mostres per ell, ens proporcionarà les dades referents a les activacions dels mapes de característiques dels diferents filtres per a la imatge analitzada. Una vegada s'han obtingut aquestes dades, s'ha d'utilitzar un nou classificador per a procedir a la finalització de la tasca. Aquest, pot ser tant una nova xarxa neuronal com un altre tipus de classificador, per exemple un SVM (Suport Vector Machine, Màquines de Suport de Vectors).

  Cal remarcar que en aquest cas la CNN utilitzada no passaria per un nou període d'aprenentatge, si no que l'aprenentatge tindrà lloc per del classificador. Adquirint així una gran importància la tasca d'escollir quin tipus de classificador és el més indicat atenent al

problema i als trets de les dades extretes per l'Extractor de Característiques que s'ha utilitzat.

- Fine-tuning: en aquest mètode a diferència de l'anterior, al final no trobem que tenim dos models, un per a l'extracció de característiques i un altre per a la classificació. Sinó que el resultat final és un model capaç tant de extraure característiques com de classificar acuradament.

  A l'igual que en el mètode que hem explicat anteriorment, la part de les xarxes preentrenades que interessa conservar és la que actua com a extractor de característiques, és a dir, les capes de convolució. Després, s'hi pot actuar de dues maneres diferents. Però la diferència més significativa és que ara si que hi ha un procés d'aprenentatge.

  La primera, consisteix en extraure la base de capes de convolució de la CNN elegida i afegir a sobre unes capes 'Densely Connected' que actuaran com a la part de classificació del model. Una vegada construïda l'estructura del que serà el nou model, abans de començar amb l'entrenament, es procedeix a la congelació de les capes de convolució, cosa que fa que durant el primer període d'aprenentatge al que serà exposat el model no es modifiquin els pesos d'aquestes capes, actuant així com a extractores de característiques. Això, és degut a que primer ens interessa aconseguir que el bloc de classificació comenci a aprendre a diferenciar entre els diferents grups d'imatges. Ja que en el suposat cas de no congelar-les o de no modificar els seu rati d'aprenentatge, al començar les primeres iteracions de l'aprenentatge s'obtindria un valor de la funció objectiu tan dolent que provocaria una actualització massiva de tots els pesos de la xarxa, perdent així l'avantatge que suposava utilitzar una xarxa preentrenada.

  Una vegada s'ha entrenat el classificador, es descongelen les últimes capes del bloc de convolució i es torna a realitzar un altre procés d'aprenentatge on ara, com la funció objectiu ja no dona uns valors tant alts, la xarxa se centra en adaptar els filtres de les últimes capes de convolució a la casuística del nou problema i en acabar de afinar l'actuació del classificador, augmentant així l'efectivitat del model.

  La segona manera possible de fer-ho és bastant similar a la primera, ja que la diferència radica en que en comptes d'haver-hi dues etapes d'aprenentatge hi ha sols una, on per tal d'evitar el problema de que es modifiquen molt el valors dels pesos de les capes de convolució reaprofitades, es posa un rati d'aprenentatge (learning rate) molt baix a aquestes i un de més alt a les capes 'Densely Connected'. Així, s'afavoreix el reaprofitament de les capes de convolució , que no variaran molt, s'afinaran ajustant-se per a les representacions més abstractes per al problema en qüestió, i l'entrenament de les capes de classificació.

3.2.6.1.1 AlexNet

Aquesta és la primera xarxa neuronal de convolució que s'hi va provar durant el desenvolupament del projecte, degut a que al començament s'hi van reaprofitar unes quantes coses d'un anterior Treball de Fi de Grau que és va desenvolupar en MATLAB, i que també tractava sobre la classificació dels mateixos plans ecogràfics però amb unes altres tècniques.

A la Figura 3-12 hi observem l'estructura de la xarxa AlexNet. La qual, es troba formada per un bloc inicial de 6 capes de convolució que tenen intercalades tres capes de 'MaxPooling' i dues de 'Cross Channel Normalization', per a la detecció dels patrons i el bloc final de 'Fully Connected', que es troba conformat per tres capes de 'Fully Connected' i dues de 'Dropout', que és el que actua com a classificador. Per tant, ens trobem amb una xarxa relativament petita (15 capes) en comparació amb moltes de les altres que s'han entrenat amb ImageNet.

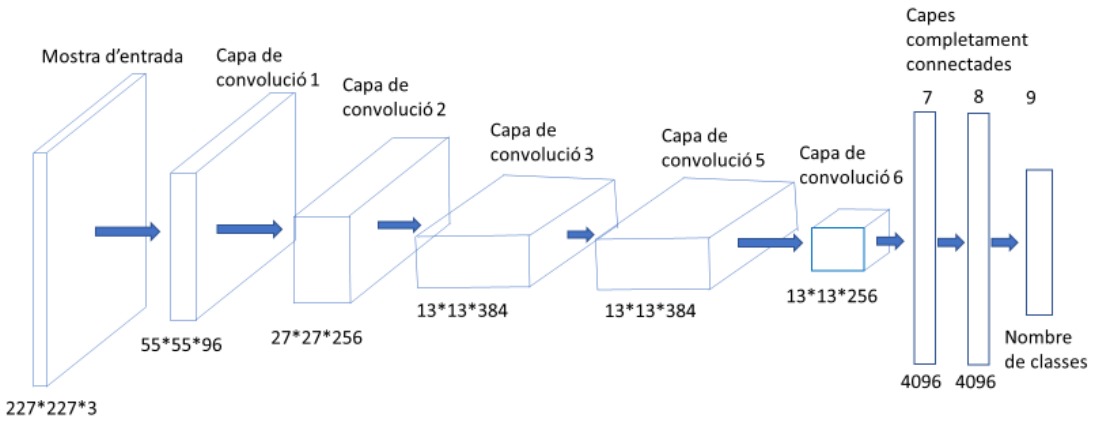


Figura 3-12. Estructura de la CNN AlexNet (Font: Pròpia)

3.2.6.1.2 VGG19

Una vegada agafada l'experiència de com treballar amb xarxes neuronals de convolució amb MATLAB, vàrem canviar a Python pels motius ja explicats. Entre les CNN que s'hi troben disponibles a Python la primera que vàrem utilitzar va ser la VGG19, la qual presenta unes dimensions més grans que l'anterior, i per tant, un nivell d'abstracció més alt també.

La CNN VGG19 també presenta dos blocs, el primer, de capes de convolució, que actua com a extractor de característiques i detector de patrons i el segon, de capes totalment connectades, que actua com a classificador.

La composició del segon bloc és la mateixa que en el cas de AlexNet. Això, es degut a com que ambdues han sigut dissenyades per al mateix conjunt de dades (ImageNet), i per tant, el nombre de paràmetres i capes de la part de classificació ha sigut optimitzada per aqueix volum determinat de dades.

En canvi, en el bloc de convolució és on s'hi troben les diferències entre una i l'altra, ja que per a l'elaboració d'aquest model s'hi varen dissenyar un bloc de convolució dividit en cinc trams, tots ells seguits d'una capa de 'MaxPooling', cosa que fa que hi hagi un conjunt total de 26 capes. Els dos primers trams es troben formats per dos capes de convolució cadascun, i els altres tres trams per quatre capes de convolució. Això, permet un major nivell d'abstracció per part del model. També, cal remarcar que en tots cinc trams s'ha utilitzat 'ZeroPadding' per evitar el fenomen de reducció que provoquen les convolucions a la imatge.

3.2.6.1.3 InceptionV3

La CNN InceptionV3 va ser la última en utilitzar-se durant el desenvolupament del projecte. Aquesta, presenta grans diferències amb les altres dues utilitzades, ja que durant el bloc de convolució va concatenant capes de convolució (amb 'ZeroPadding' per evitar el fenomen de reducció) amb capes 'MaxPooling' fins al moment d'haver passat de l'entrada de 299*299*3 a la sortida de 8*8*2048, a la qual se li aplica un 'AveragePooling' per obtenir un vector de 2048 atributs per a cada mostra. La seva llargada també presenta grans diferències, ja que consta de 313 capes, 310 referents al bloc de convolució i les últimes una per adequar les dades per al classificador, i l'altra la capa 'Fully Connected' que actua de classificador. Aquest vector és l'entrada al classificador d'aquest model, que es troba conformat bàsicament per una única capa de 'Fully Connected', la capa que té l'activació 'softmax' que troba la probabilitat de pertinença a les diferents classes.

3.2.6.2 CNN a mida

En molts casos pot resultar interessant construir el teu propi model, per això, s'ha de valorar si el conjunt de dades de que es disposa es suficient com per a l'obtenció d'un que esdevingui una bona representació del conjunt de dades. Un altre dels punts clau que s'han de tenir molt en compte es l'elecció del nombre de capes que s'utilitzaran i el nombre de paràmetres de cadascuna de les capes,

perquè el que s'ha d'intentar evitar a tota costa és el fenomen conegut com 'Overfitting', que significa que el model no generalitzi bé per a la tasca de reconeixement i classificació dels diferents plans ecogràfics.

El problema del 'Overfitting' ocorre en tots els problemes d'aprenentatge automàtic, tant aquells on s'utilitza una xarxa preentrenada com en els casos en que se'n crea una de nova, ja que com tota la tasca d'aprenentatge està per realitzar, el model és més sensible a aquest fenomen. I la clau s'hi troba en la tensió existent entre l'optimització, entesa com el procés d'ajustar el model per a aconseguir un compliment més eficaç de la tasca en referència a les mostres d'entrenament, i la generalització del model, referit a com de bé respon el model una vegada entrenat front mostres que no ha vist mai.

La metodologia utilitzada per a la resolució d'aquest problema es basa en inicialment elaborar un model amb una estructura més o menys simple, i realitzar un procés d'aprenentatge d'aquest model amb les mostres d'entrenament. Una vegada s'han complert unes quantes iteracions del procés s'observa si realment el model està generalitzant, classificant les mostres de validació correctament, o en canvi està aprenent diferenciar entre les dades d'entrenament però davant mostres mai vistes no funciona correctament. En relació al resultat que s'observi, es modificarà d'una manera o una altra el model.

Per reduir el 'Overfitting' s'ha de limitar la complexitat del model, tant reduint les dimensions del mateix, com el nombre de paràmetres o canviant el tipus d'estructura. Això es degut a que la presència d'un nombre molt elevat de paràmetres porta a una situació on la relació paràmetres / mostres d'entrenament s'hi troba descompensada, és a dir, hi ha suficients com per a poder aprendre les característiques de cadascuna de les imatges. Això, significa que el model resultant és incapaç de generalitzar per a mostres no vistes, ja que en comptes d'aprendre patrons propis de cada classe d'imatges, ha aprés els patrons propis de cada mostra. Per contra, si esdevé 'Underfitting', que significa que el model no ha sigut capaç de modelar tots els patrons rellevants que caracteritzen els diferents plans, la metodologia d'actuació seria la contrària, augmentar la complexitat, dimensió i/o paràmetres de la xarxa.

En el cas de les xarxes neuronals de convolució que es dissenyen a mida no s'hi contempla el seu ús com a Extractor de Característiques, ja que al definir el model tots els pesos s'inicialitzen amb valors aleatoris, amb la qual cosa, com que no detecta patrons definits i representatius el seu ús resultaria bastant ineficaç, i el seu ús una vegada entrenada també esdevé innecessari, ja que el classificador del model ja s'hi troba optimitzat per a la identificació d'aquest tipus de mostres. Per tant, el seu ús queda reduït a l'entrenament del model amb la part de les mostres definida per a entrenar, fins comprovar satisfactòriament l'efectivitat del model. Una vegada trobada l'estructura òptima del model, es procedeix al seu entrenament amb la totalitat del conjunt de dades, per obtenir així el model a comercialitzar.

3.2.7. Altres classificadors utilitzats

Aquest es tracta d'un tipus de classificador que en origen es binari lineal, és a dir, troba una única frontera de separació entre dues classes, representada per un hiperplà. Per tant, la tasca que du a terme és la de definir la frontera lineal entre els dos grups a partir de es mostres utilitzades per a l'entrenament.

Defineix un hiperplà que troba la frontera a partir de les mostres d'entrenament. Per a això, sols es tenen en compte el vectors de suport de cada classe, i amb la condició de que aquests vectors de suport s'elegeixen de manera que la distància entre els plans que els contenen sigui màxima. Així, es troba la regió més amplia de l'espai de característiques que separa les dues classes i que es troba buida de mostres. Per tal de trobar aquest hiperplà que maximitza el marge, s'aplica un algoritme d'optimització.

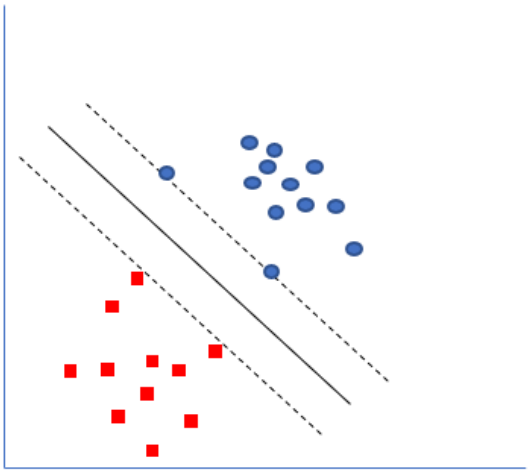


Figura 3-13. Mostra de les Imatges del tipus NT (Font: Pròpia)

Aquest hiperplà que separa les dues classes no es troba directament, sinó que primer es defineix la regió compresa per dos hiperplans que contenen els vectors de suport de cada classe de mostres, i per tant, a partir d'aquests, es troba el hiperplà intermedi, que si que representa la solució. Aquesta solució, permet obtenir el màxim marge de seguretat en la classificació de noves mostres.

El principal problema es degut a que en la gran majoria dels casos, les distribucions de les mostres no són linealment separables. Per tant, les SVM incorporen una sèrie d'algoritmes per trobar tant les

transformacions espacials de les característiques de les mostres, com els canvis a espais de dimensions superiors per tal d'aconseguir una representació de les mostres linealment separable.

En aquells casos on el problema a resoldre no es tracta d'una classificació binaria, les SVM defineixen un nombre d'hiperplans definit per l'Equació 3.7. Procés que simbolitza la combinació de tants classificadors binaris com faci falta per definir les fronteres entre tots els diferents grups.

$$\frac{\boldsymbol{n_{classes}}*(\boldsymbol{n_{classes}}-\mathbf{1})}{\mathbf{2}} \qquad \textbf{(Eq. 3.7)}$$

Per al procés de classificació de les noves mostres, s'utilitza la distribució de distàncies obtinguda de les mostres d'entrenament per estimar la probabilitat de les noves mostres. Per tant, per estimar la probabilitat de pertinença a un grup o un altre, empra la distància entre les mostres a classificar i l'hiperplà que separa dos grups. Com aquesta distància representa la proximitat de les noves mostres a cada classe, a partir de les distàncies trobades obté la seva predicció i les probabilitats de pertinença a cadascuna de les classes.

3.2.8. Protocols de Validació

En les tasques de classificació automàtica, s'entén com a validació el procés mitjançant el qual es comprova com de bé ha après a generalitzar a partir de l'entrenament amb un conjunt de dades, és a dir, a reconèixer correctament la classe d'unes dades que no ha vist.

La validació simple, que consisteix en partir el conjunt de dades en dos parts, una per a l'entrenament i l'altra per a la validació porta associat un problema degut a que el funcionament del model pot variar molt en funció de la partició d'exemples que s'haja realitzat, segons la partició s'obtindran uns resultats o uns altres. Amb l'objectiu de minimitzar aquest efecte, es sol utilitzar la Validació Creuada (coneguda en anglès com cross-validation). El funcionament d'aquest tipus de validació consisteix en dividir el conjunt de dades en K subconjunts, i realitzar K proves utilitzant per a cada una un subconjunt com a test i els altres com a entrenament. Després, es calcula la mitja aritmètica i la desviació estàndard, que aporten un major coneixement sobre el comportament real del sistema.

Per a realitzar una avaluació més exhaustiva dels resultats, es calculen una sèrie de mesures d'avaluació a partir de la matriu de confusió. Aquests són la Sensibilitat i la Especificitat, en quant a la sensibilitat, correspon als exemples positius ben classificats sobre el total d'exemples positius.

$$Sesibilitat = \frac{VP}{VP + FN} \qquad \textbf{Equació 3.8}$$

On VP es correspon a aquelles mostres d'una classe correctament classificades i FN a aquelles mostres de la mateixa classe que no s'han classificat correctament.

En quant a l'Especificitat, indica l'efectivitat del classificador per identificar la classe negativa, i es calcula com veiem a l'Equació 3.9:

$$Especificitat = \frac{VN}{VN + FP} \qquad \textbf{Equació 3.9}$$

On VN representen els casos correctament classificats com negatius, i FP aquells classificats incorrectament com positius.

3.3. Preprocessat

El pas previ a tot problema de Visió per Computador és el preprocessat de les imatges, per així eliminar tota informació irrellevant del conjunt de dades, i obtenir uns millors resultats en la tasca a realitzar.

El flux de treball utilitzat per al preprocessat de les imatges es troba il·lustrat en la Figura 3-14. Flux de treball inspirat en el realitzat en el projecte "Biometric Plane Classification in Fetal Ultrasound Scan", projecte corresponent al Treball de Fi de Grau de la estudiant del Grau en Enginyeria Biomèdica de la UPC Safae Bendali [19]. Projecte utilitzat com a punt de partida per a la realització del present, i del qual s'ha reaprofitat tot referent a la part de preprocessat de les imatges. Ja que algunes són les mateixes, i les altres segueixen la mateixa casuística. Per tant, tot el que s'explica en aquest apartat referencia al codi elaborat al treball citat, que en gran part s'ha reutilitzat, però que li hem modificat diverses coses per trobar un resultat més acord amb els nostres interessos.

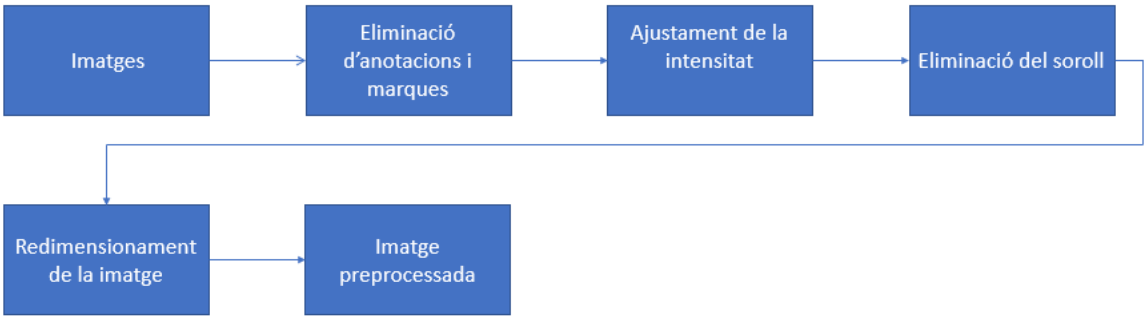


Figura 3-14. Flux de treball del preprocessat (Font: Pròpia)

Un dels aspectes clau quan es treballa amb Conjunts de Dades amb una mida considerable és que es fa necessària una tasca d'automatització del preprocessat, així no s'ha de realitzar mostra a mostra amb l'elevat temps que portaria associat aquesta pràctica. Amb l'objectiu de trobar el mètode apropiat, es varen realitzar diferents probes amb mostes de cadascuna de les classes per validar el resultat del preprocessat. Llavors, el procediment finalment adoptat es aplicat a totes les imatges per dur-lo a terme de manera automàtica.

3.3.1. Eliminació d'anotacions i marques

En primer moment es procedeix a l'eliminació de les anotacions de text i altra informació irrellevant present a les imatges ecogràfiques. La mostra de la Figura 3-15 exposa algunes marques i anotacions resultats de l'ús de calibradors electrònics per al mesurament fetal. Aquests atributs no aporten informació clínica rellevant i dificulten una possible tasca de classificació. A la mostra del tipus BPD mostrada en la Figura 3-15 s'hi observen píxels de color blau i groc que deurien ser eliminats de la imatge. A més, també s'observa la presència de píxels que corresponen a texts, aquests encara que més difícils, també s'han d'eliminar o reduir.

Com que els píxels elegits per ser eliminats es troben entre regions anatòmiques del fetus, es necessari una reconstrucció de la imatge per tal de mantenir les característiques d'aquestes Estructures Anatòmiques Clau tan immutables com sigui possible en les representacions Biomètriques dels plans fetals.

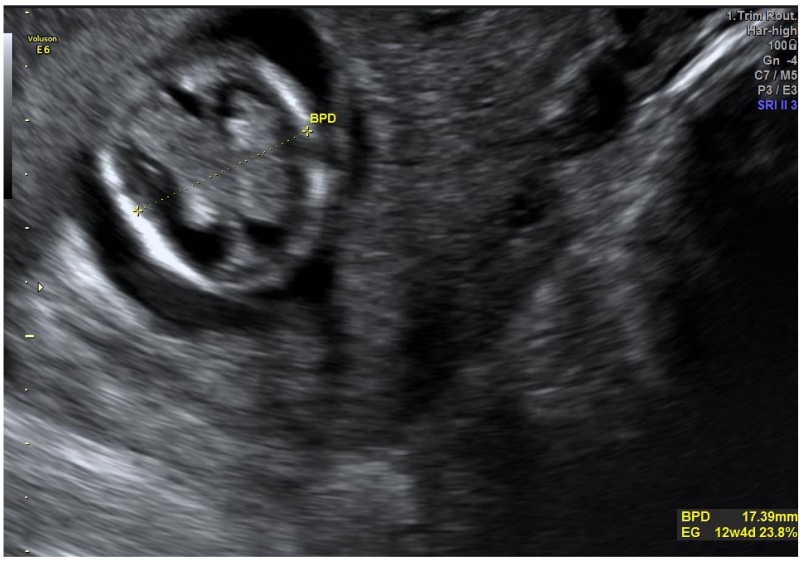


Figura 3-15. Mostra original d'una Imatge on es veuen les anotacions i marques a eliminar (Font: Pròpia)

L'eliminació dels píxels de colors irrellevants de la imatge es va realitzar mitjançant l'ajuda del MATLAB App del llindar de color per seleccionar valors màxims i mínims per a histogrames de píxels de color amb canals RGB. Trobant així aquells píxels que es volen substituir Figura 3-16.

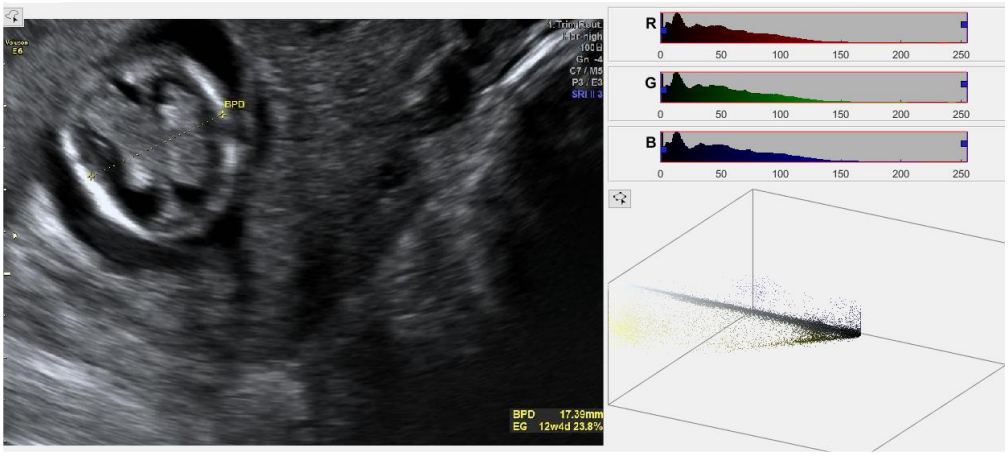


Figura 3-16. Una mostra amb la informació dels canals RGB (Font: Pròpia)

La funció "RemoveMarks.m" (mirar Annex A1) es va desenvolupar per a les eliminacions d'anotacions a color. Ona vegada definits els valors màxims i mínims per a les configuracions dels histogrames de RGB, la idea es trobar la localització d'aquests píxels que compleixen la condició i substituir-los per la mitja dels valors dels seus píxels veïns. Per a això, s'utilitza un filtre fspecial de mida [20 20] que sols operarà en les ubicacions seleccionades.

Els resultats d'executar la funció sobre una mostra del tipus BPD es representada a la Figura 3-17. I d'una forma similar, per a una mostra d'un tipus CLR, i del tipus NT, en les Figures 3-18 i 3-19 respectivament. Comprovada l'eficàcia de la funció sobre les marques de color i les anotacions de text, es va decidir per mantenir la funció en el seu estat original.

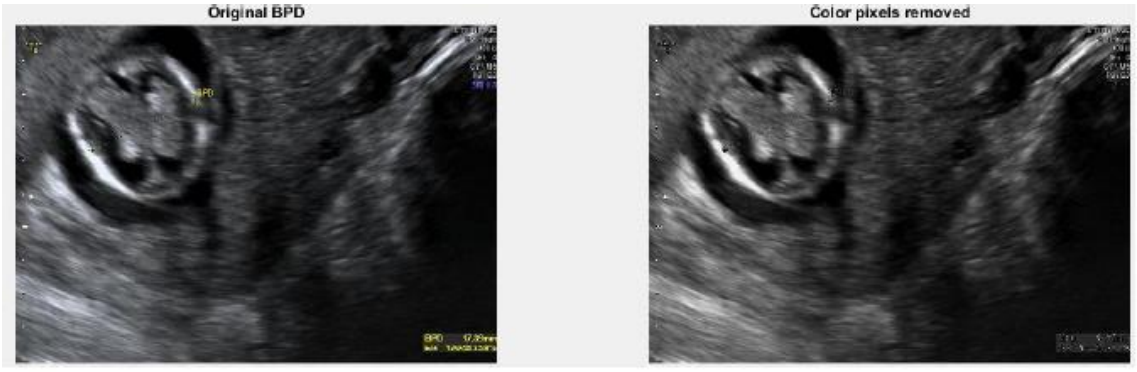


Figura 3-17. Comparació d'una BPD original amb el resultat de l'eliminació d'anotacions (Font: Pròpia)

Figura 3-18. Comparació d'una CRL original amb el resultat de l'eliminació d'anotacions (Font: Pròpia)

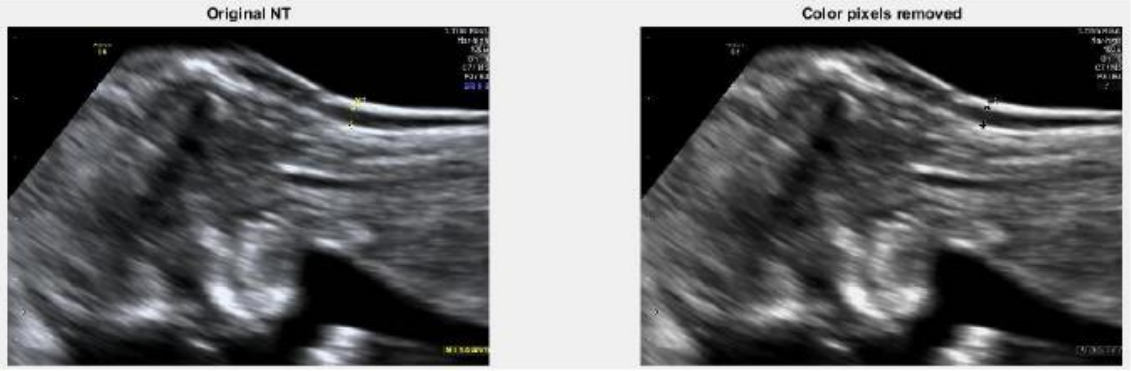


Figura 3-19. Comparació d'una original amb el resultat de l'eliminació d'anotacions NT (Font: Pròpia)

3.3.2. Ajustament de la intensitat

Amb l'objectiu de realçar la intensitat de les imatges, es va utilitzar tot el codi de que disposàvem referent a aquesta part, realitzant les comparacions entre les tècniques d'ajustament del contrast i d'equalització de l'histograma per determinar el mètode més apropiat per a treballar amb imatges ecogràfiques, i constatar que les decisions que es van prendre a [19] eren les correctes. L'ajustament del contrast és originalment utilitzat per motius de visualització per realçar la representació contingut de la imatge. Això, va portar a comprovar i comparar l'equalització de l'histograma i l'ajustament de la intensitat en diverses imatges de mostra. El procés d'equalització de l'histograma té lloc sobre el propi histograma de la imatge, redistribuint els píxels per a igualar el nombre de píxels en cadascuna de les intensitats de l'escala de grisos segons els llindars establerts. Es tracta d'una transformació no lineal de la imatge que altera els valors dels píxels, mentre que l'ajustament de la intensitat balanceja les intensitats de la imatge cobrint tot el rang. La configuració per defecte calcula el 1% dels valors de píxel tant per baixa intensitat com per alta. Trobant un balanceig lineal de la imatge original, on cap altra informació a part de les intensitats originals dels píxels es perd.

Les comprovacions prèvies es varen realitzar sobre mostres del nostre conjunt de dades per a sospesar com realçar satisfactòriament les imatges ecogràfiques abans de construir l'algoritme final de preprocessat per a tot el conjunt. Les diferents comparacions entre l'ajustament d'intensitat i l'equalització d'histograma es mostren en la Figura 3-20 per a una mostra del tipus BPD, Figura 3-21 per a CRL i 3-22 per a NT.

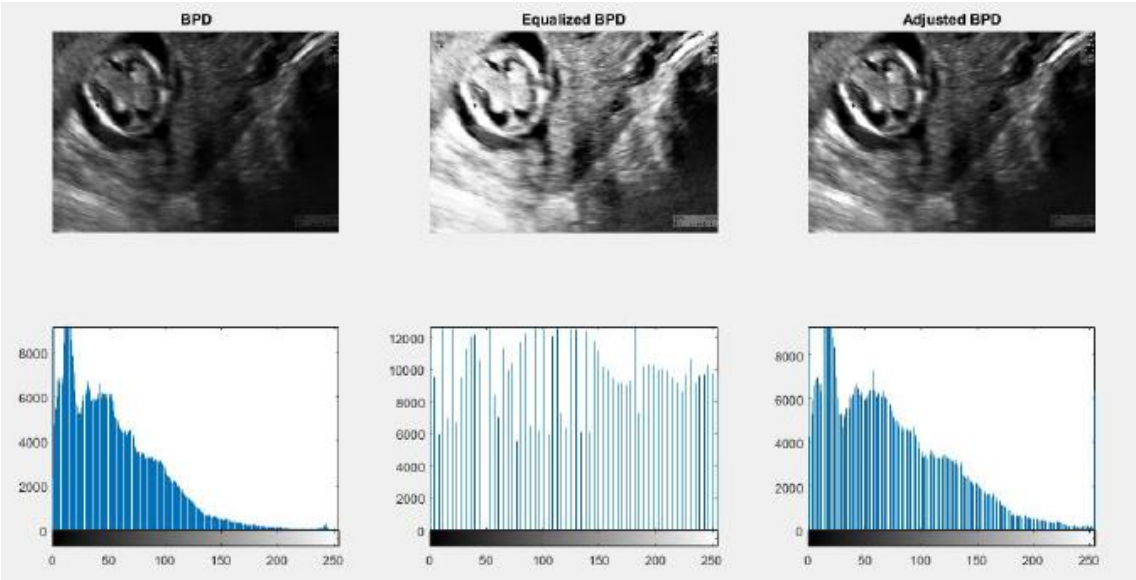


Figura 3-20. Comparació de la intensitat després del realçament a una BPD (Font: Pròpia)

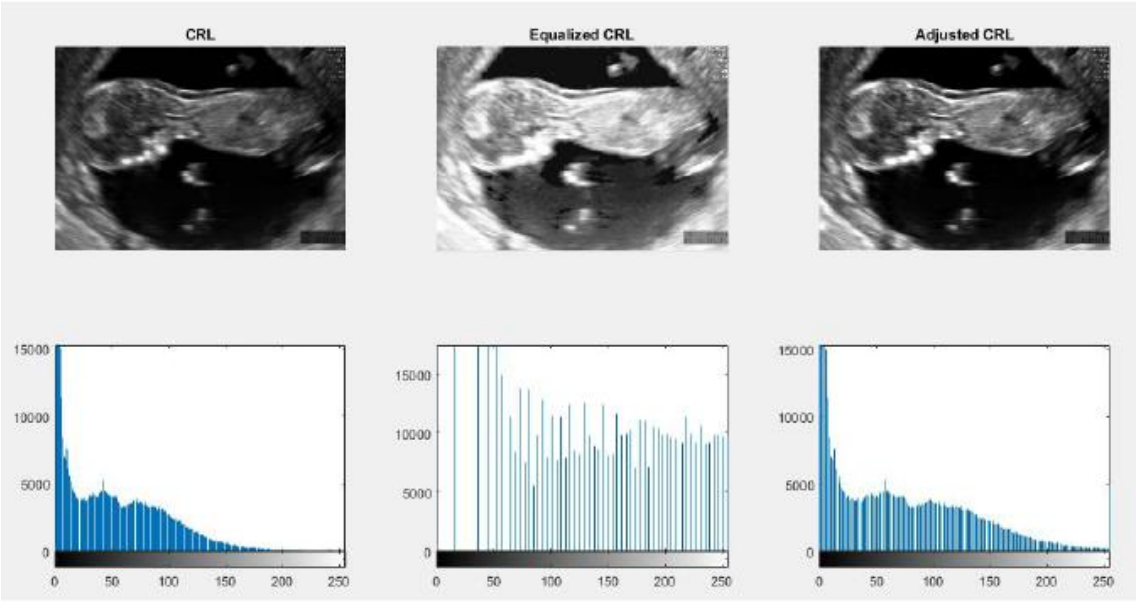


Figura 3-21. . Comparació de la intensitat després del realçament a una CRL (Font: Pròpia)

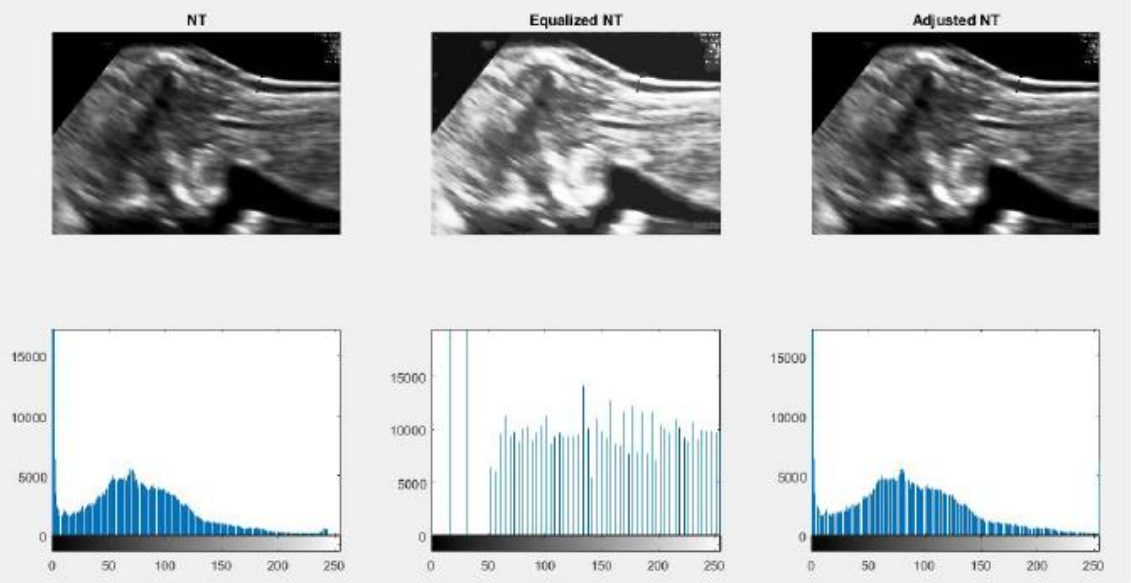


Figura 3-22. . Comparació de la intensitat després del realçament a una NT (Font: Pròpia)

Les primeres observacions revelen que l'equalització de l'histograma proporciona imatges amb més brillantor, en canvi, això també produeix un augment del soroll de la imatge, efecte no desitjat que no resulta compatible amb l'objectiu del preprocessat. Els resultats de les comprovacions mostren que l'ajustament d'intensitat dona com a resultat unes millors imatges, realçant les estructures anatòmiques clau, sense amplificar artefactes no desitjats. En conclusió, el mètode d'ajustament d'intensitat serà el que finalment serà aplicat per aconseguir un realçament de la imatge com a part de la fase de preprocessat.

3.3.3. Eliminació del soroll

El següent pas és l'eliminació del soroll. Entre tots els tipus de filtre, es va elegir un filtre de mediana (típic filtre per a la reducció del soroll) per comprovar l'efecte que hi tenia sobre les mostres d'imatges ecogràfiques la utilització de diferents kernels, on els més rellevants entre els comparats van ser el [6 6] i [20 20]. Per avaluar quin d'aquests és el més idoni sols cal observar detingudament la Figura 3-23.

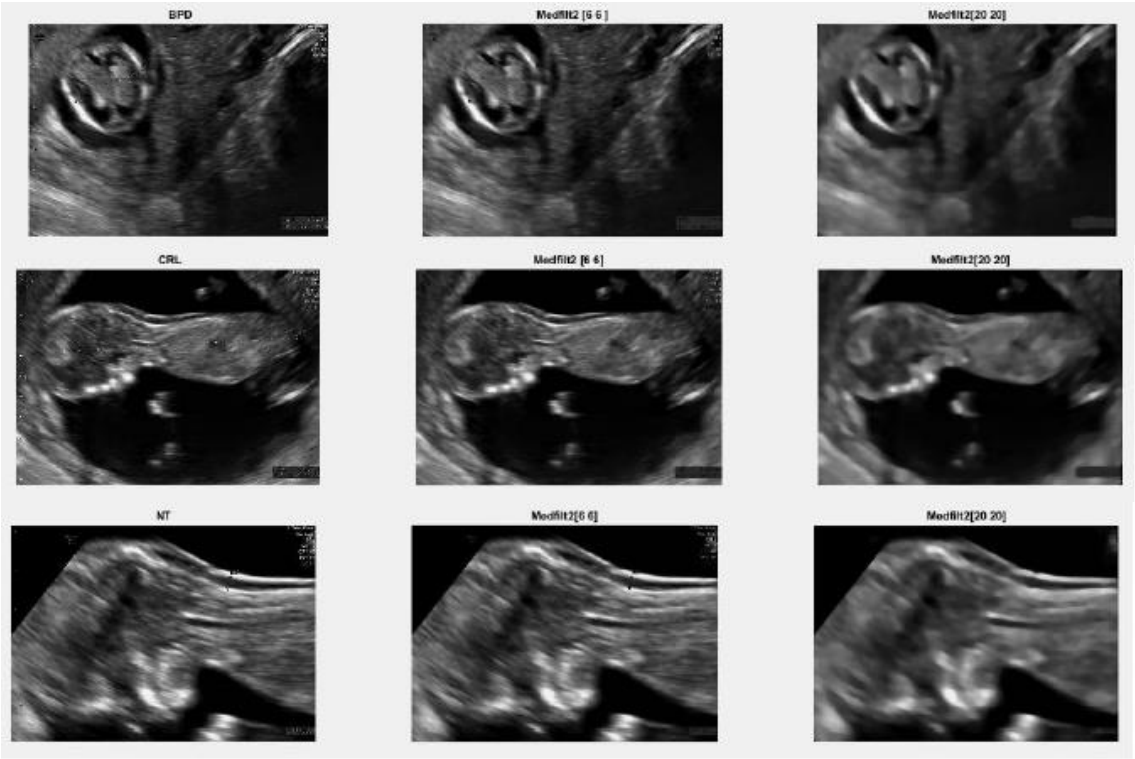


Figura 3-23. Comparació dels resultats d'aplicar els diferents mides de kernel a diferents mostres (Font: Pròpia)

Després de comprovar els resultats que s'han obtingut en les mostres de diferents classes, es va concloure que amb l'aplicació d'un filtre de mediana amb un kernel de [6 6] es trobava el millor resultat per a les nostres imatges ecogràfiques. Aquest mètode adoptat suavitza el soroll restant (píxels de textos, fletxes i marques a color) que varen escapar a la selecció en els apartats anteriors. La intensitat dels píxels també es suavitza i es guanya més uniformitat.

3.3.4. Redimensionament de la imatge

L'última etapa del procés de preprocessat és el redimensionament de les imatges. Totes les imatges són redimensionades a dues mides diferents, una de 227*227 i una altra de 299*299 per a permetre la seva utilització en les diferents experiments amb xarxes neuronals de convolució, i evitar futurs redimensionaments que podrien provocar la pèrdua . La funció de MATLAB "ImPrep.m" (veure l'Annex A1) és la encarregada de dur a terme l'ajustament de la intensitat de la imatge, l'aplicació del filtre mediana per a la reducció del soroll amb un kernel de [6 6] i el redimensionament de la imatge.

Per tant, al final ens trobem amb tres diferents conjunts de dades amb de les mateixes mostres. Un amb les imatges originals obtingudes a través de l'Hospital Sant Joan de Deu i les extretes d'internet per la Safae Bendali, un altre amb les imatges ja preprocessades i amb una mida de 227*227, i un últim també amb les imatges preprocessades, però amb una mida diferent, 299*299.

3.4. CNN com a Extractor de Característiques

En aquests apartats es procedirà a l'explicació de com s'han realitzat els diferents experiments, i el perquè de les decisions preses. Primer s'hi començarà exposant tot allò referent als experiments on s'han utilitzat CNN com a Extractor de Característiques i un SVM com a classificador.

3.4.1. AlexNet

Per a la utilització de la CNN preentrenada AlexNet, es va haver d'aplicar una modificació de les dimensions de les imatges, perquè la grandària de les originals era de 960*720*3 píxels (imatges en color) i les dimensions exigides per la xarxa per a les dades d'entrada era de 227*227*3. Per a l'ús d'aquesta xarxa Mathworks aconsella no realitzar una normalització de les imatges [14].

```
newIm= imresize(newG, [229 229 3]);
```

Una vegada es disposa del conjunt d'imatges amb la mida correcta i minuciosament separat en carpetes segons la classe de la imatge, ja es pot procedir al desenvolupament de la tasca en qüestió. Per a això, utilitzem la funció `'imageDatastore'`, la qual, especificant-li la carpeta on es troba tot el conjunt de mostres degudament separades, crea una estructura que conté les rutes de totes les imatges i la etiqueta corresponent al grup de cadascuna.

El següent pas consisteix en procedir a la partició de quines imatges seran utilitzades per a l'entrenament, i quines per a la validació de l'entrenament. Per això, s'ha realitzat una partició mitjançant la funció `'splitEachLabel()'` de més o menys el 80% de les mostres per a entrenament (406 mostres per classe), i el 20% restant per a la validació (100 mostres per classe).

En el cas concret de MATLAB, al trucar un model entrenat, ja importa directament els valors dels pesos de la xarxa resultants del procés d'aprenentatge al que va ser sotmès. Per tant, el que queda és l'elecció de fins a quina capa de la xarxa es vol utilitzar per a l'extracció de característiques.

S'ha decidit utilitzar totes les capes exceptuant la última, la que presenta l'activació 'softmax' que retorna probabilitats de pertinença. Per tant, queda un model que actua com extractor de característiques amb una estructura formada per tot el bloc de convolució d'AlexNet i les dos primeres capes del tipus 'Fully Connected', que proporcionaran com a sortida un conjunt de 4096 propietats per a cada mostra. Això s'aconsegueix utilitzant la funció `'activations'`, a la que se li especifica l'estructura de la xarxa a emprar amb els seus pesos, les mostres d'entrada i la capa que serà l'última, i retorna les activacions produïdes al llarg de les capes per les dades de les mostres d'entrada. A continuació trobem el codi MATLAB que permet realitzar allò que s'acaba d'explicar:

```
layer = 'fc7';

trainingFeatures = activations(net,trainingImages,layer);

testFeatures = activations(net,testImages,layer);
```

Una vegada extretes les propietats de les mostres, arriba el moment d'entrenar el classificador seleccionat i avaluar el nivell d'efectivitat. Com ja s'ha especificat en apartats anteriors, en aquells casos que s'utilitza una CNN com a extractor de característiques el classificador que s'utilitzaran seran les Màquines de Vectors de Suport (SVM). Per tant, es procedeix a l'entrenament el classificador proporcionant-li les mostres en format de vector de propietats i les seves respectives etiquetes.

```
classifier = fitcecoc(trainingFeatures,trainingLabels);
```

Una vegada entrenat el classificador amb el lot de mostres destinat a l'entrenament, el següent pas és l'avaluació del classificador. En aquest pas se li dona com a entrada del model entrenat els vectors de propietats de les mostres de validació, i ell proporciona com a sortida les prediccions de les etiquetes que creu que corresponen a cada mostra. I per avaluar si les ha classificat correctament o no es calcula l'exactitud comparant les etiquetes correctes amb les predites.

```
predictedLabels = predict(classifier,testFeatures);

accuracy = mean(predictedLabels == testLabels)
```

Una vegada comprovat el correcte funcionament del classificador l'últim pas seria entrenar-ne un de nou, però aquesta vegada amb totes les mostres (el 100%), i el model resultant junt a la part de 'Feature Extraction' seria el que es comercialitzaria.

3.4.2. VGG19

En aquest moment es va canviar de llenguatge de programació, degut a com ja s'ha explicat anteriorment, Python disposa de nombroses llibreries amb moltes xarxes neuronals de tot tipus ja implementades i llestes per a utilitzar.

Per tant, els passos en aquest experiment varen ser bastant similars als de l'anterior, exceptuant que a Python no hi ha cap funció com la de `'imageDatastore'` del Matlab, cosa que va obligar a realitzar tot el procés de preparació de les dades i adequació per a la seva utilització a mà. La partició del conjunt de dades per a entrenament i validació s'ha realitzat d'acord amb les especificacions explicades a l'apartat anterior.

Degut als requisits de la xarxa VGG19, les imatges havien de tenir les dimensions 225*225*3. Per aconseguir imatges d'aquestes mides el que es va realitzar va ser utilitzar les mateixes que per a la xarxa AlexNet, però eliminant dos píxels d'alçada i dos d'amplada. En quant a la necessitat o no d'un procés previ de normalització de les imatges, es varen trobar informacions contradictòries que motivaren la realització d'un estudi de comprovació per veure quina normalització resultava més aconsellable, o si era millor treballar sense normalització.

Una vegada amb les dades ja preparades per al seu ús, arriba el moment d'importar la xarxa neuronal de convolució VGG19 i els pesos resultants del seu entrenament amb el conjunt de dades 'ImageNet'. I de definir quin tram de la xarxa és el que es vol utilitzar per a la tasca d'extracció de característiques, que al igual que en l'anterior experiment, agafem fins a la capa justament anterior a l'última, la que té l'activació 'softmax' (actua com a classificador).

```
from keras.applications import VGG19
from keras.models import Model
base_model = VGG19(weights='imagenet')
model = Model(inputs=base_model.input, outputs=base_model.get_layer('fc2').output)
```

Figura 3-24. Codi per a trucar VGG19 (Font: Pròpia)

Una vegada amb l'extractor de característiques preparat, es procedeix a l'extracció utilitzant com a entrada les matrius corresponents a les diferents imatges, separadament primer les d'entrenament i després les de validació.

```
trainingFeatures = model.predict(trainingImages)
testFeatures = model.predict(testImages)
```

Figura 3-25. Codi per a l'extracció de característiques (Font: Pròpia)

Per tant, ara sols queda entrenar el classificador (SVM) amb les mostres d'entrenament i procedir a l'avaluació de la seva eficàcia.

```
svc = SVC(C=100, kernel='poly', degree=2)
svc = svc.fit(trainingFeatures, trainingLabels)
solu=svc.predict(testFeatures)

print(accuracy_score(testLabels, solu))
print(confusion_matrix(testLabels, solu))
```

Figura 3-26. Codi per a l'entrenament del SVM i la predicció (Font: Pròpia)

Com es pot apreciar a les anteriors les línies de comandes, s'han especificat quins paràmetres havia de tenir la Màquina de Vectors de Suport, això es degut a que la funció que utilitzàvem a MATLAB ja ens trobava el classificador més òptim per a les dades de que disposava, però en canvi per a Python s'ha de realitzar un estudi de quina és la combinació més òptima de paràmetres. Per tal de trobar quins són els òptims, s'han utilitzat les línies de comandes de la Figura 3-26, on es fa un rastreig per trobar quina combinació de totes proporciona més bon resultat per a les nostres dades. I realitzant una 'Cross Validation' per reduir possibles fenòmens aleatoris.

Després de realitzar aquest rastreig, es va concloure que el classificador més òptim per al nostre cas es tractava d'aquell que hi tenia uns valors de C=100, i un kernel polinòmic de grau dos.

```
param_grid = [
  {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [10, 1, 0.1, 0.01, 0.001, 0.0001], 'kernel': ['rbf']},
  {'C': [1, 10, 100, 1000], 'degree': [2, 3], 'kernel': ['poly']}
 ]
tuning(GridSearchCV(SVC(), param_grid, cv=5, scoring='accuracy'), \
       'SVM', param_grid)
```

Figura 3-27. Codi per a trobar els paràmetres més òptims per al SVM (Font: Pròpia)

3.4.3. InceptionV3

Per acabar d'analitzar els resultats de treballar amb CNN com a Extractor de Característiques, es va optar per utilitzar la xarxa InceptionV3, també disponible a la llibreria 'Keras' de Python.

Una de les particularitats d'aquesta és la gran llargada del bloc de convolució, però també té com a particularitat la mida de les imatges que exigeix com a entrada (299*299*3). Per la qual cosa, es va haver de realitzar altra vegada un procés de redimensionament de les imatges originals, per tal d'evitar al màxim la possible aparició de fenòmens de distorsions causats per els successius redimensionaments.

Una vegada acabada la part de preparació de les dades per a treballar amb la xarxa escollida es procedeix a l'elecció de la capa del model que utilitzarem com a sortida del 'Features Extractor'.

```
from keras.applications.inception_v3 import InceptionV3
from keras.models import Model
base_model = InceptionV3(weights='imagenet')
model = Model(inputs=base_model.input, outputs=base_model.get_layer('avg_pool').output)
```

Figura 3-28. Codi per a trucar la CNN InceptionV3 (Font: Pròpia)

La CNN InceptionV3 no disposa del petit conjunt de capes del tipus 'Densely Connected' just abans de la última capa amb l'activació 'softmax', sinó que utilitza les successives capes del bloc com a reductores de la dimensionalitat, fins al punt que la capa anterior a la 'softmax', rep com a entrada 2048 finestres d'una mida de 8*8, que representen les successives identificacions i activacions dels filtres de les capes de convolució. Aquesta, és una capa 'AveragePooling' amb una finestra de mida 8*8, per tant, obté com a sortida la mitjana de cadascuna de les finestres que havia rebut en un format que ja permet la classificació mitjançant la Màquina de Vectors de Suport. La Figura 3-28 detalla aquesta transformació.

Entrada (nombre_mostres, 8, 8, 2048) → AveragePooling (8, 8) →
→ Sortida (nombre_mostres, 2048)

Figura 3-29. Esquema de la transformació que sofreixen les dades (Font: Pròpia)

El següent pas és el de passar les imatges d'entrenament i les de validació per l'extractor de característiques, la sortida del qual s'utilitza per a l'entrenament del classificador (SVM). El classificador utilitzat ha sigut el mateix que en l'apartat anterior, perquè com s'ha exposat, té els paràmetres optimitzats per al problema en qüestió.

Una vegada realitzat tot aquest procés, es procedeix a l'anàlisi i comparació dels resultats, punt que es tractarà més endavant.

3.4.4. Aplicacions extres

Les metodologies de treball anterior permeten obtenir prediccions concretes per a cada mostra, cosa que no resulta de molta ajuda quan es disposa de conjunts de dades tant amb mostres representatives de les diferents classes com de no representatives. Casuística distintiva del que s'hi trobarà el model en la vida real, una vegada es procedís a una hipotètica comercialització. Ja que, com ja s'ha explicat, el que interessa és passar-li totes les imatges preses durant tota una sessió d'ecografia.

La manera més eficaç d'aconseguir que el programa sigui capaç de distingir entre aquelles imatges que si que corresponen a un dels plans i classificar-les al seu grup corresponent, i aquelles que no, és utilitzant una variant de la comanda utilitzada fins ara per trobar les prediccions amb la SVM. Aquesta altra comanda especifica al classificador que en comptes de proporcionar com a sortida la classe predita, que proporcioni la probabilitat de pertinença a cada classe que ha trobat per a la mostra.

Després de realitzar un estudi exhaustiu de les probabilitats assignades per a les imatges classificades, s'ha trobat que sempre que classifica una mostra correctament la probabilitat de pertinença a eixa classe és superior al 55%.

```
resultat=svc.predict_proba(testFeatures)
```

Figura 3-30. Codi per a la predicció de les etiquetes de les mostres que formen "testFeatures" (Font: Pròpia)

Per tant, si es defineix un llindar del 55% de probabilitat per a la classificació de la imatge, el programa seria capaç de rebutjar tant aquelles imatges no representatives de cap dels plans escollits com les que no s'han identificat correctament.

Aquesta hipòtesi es va comprovar satisfactòriament, observant com amb aquesta modificació el programa era capaç d'eliminar aquelles imatges que no formaven part de cap dels grups, però eliminant aquelles que encara tot i pertànyer a un dels grups definits, no van ser classificades correctament, augmentant així l'efectivitat del model a cost de perdre algunes imatges que realment si que eren del nostre interès.

3.5. Classificació amb CNN per Transferència de l'Aprenentatge

En aquest apartat s'explicaran tots els passos seguits en els tres experiments realitzats per a la metodologia també coneguda com 'Fine-tunning', on s'utilitza el bloc de convolució prèviament entrenat de les CNN especificades per a la creació d'una nova CNN que sols ha d'ajustar la part de classificació i molt poc les capes de convolució, per així adequar-les a les representacions més abstractes pròpies del nostre conjunt de dades. Aquesta transferència de coneixement d'un model a un altre es tractarà d'aprofitar al màxim, i fer la explicació de com s'ha aconseguit el més clar possible.

3.5.1. AlexNet

Com ja s'ha explicat en anteriors apartats, es va començar treballant en MATLAB, per tant, la primera CNN utilitzada per a realitzar aquest experiment també va ser AlexNet.

En aquest experiment s'hi van reutilitzar les mostres que ja s'havien adequat per a la seva utilització en l'experiment de 'Features Extraction' amb la CNN AlexNet, ja que les exigències d'entrada són les mateixes, així com també s'hi va utilitzar la mateixa partició de mostres per a entrenament i validació.

En aquest cas es reutilitzarà una part de la xarxa entrenada per a construir la nova, aquesta part serà la que faja efectiva la transferència de coneixement. La divisió realitzada separa tot el bloc de convolució junt a les dues primeres capes de 'Fully Connected' de la última capa, la 'Fully Connected' amb activació 'softmax', que és la que actua de classificador i s'hi troba configurada per a 1000 classes diferents a les nostres.

```
layersTransfer = net.Layers(1:end-3);
numClasses = numel(categories(trainingImages.Labels))
layers = [
    layersTransfer
    fullyConnectedLayer(numClasses,'WeightLearnRateFactor',20,
        'BiasLearnRateFactor',20)
    softmaxLayer
    classificationLayer];
```

Figura 3-31. Codi MATLAB per condicionar la CNN al problema (Font: Pròpia)

Per a la creació de la nova xarxa a partir de les capes extretes, s'ha de definir un nou classificador. Aquest, serà una capa del tipus 'Fully Connected' amb activació 'softmax', però configurada per a discernir entre tres grups. Aquí la nomenclatura abans explicada varia, ja que per a Python, aquesta última capa seria definida amb sols una línia de codi on s'especifica l'activació, i per tant ja actuaria com a classificadora.

```
options = trainingOptions('sgdm', ...
    'MiniBatchSize',20, ...
    'MaxEpochs',7, ...
    'InitialLearnRate',1e-4, ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',3, ...
    'ValidationPatience',Inf, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

Figura 3-32. Codi MATLAB per especificar les opcions d'entrenament de la CNN (Font: Pròpia)

Un punt clau per assegurar el correcte aprenentatge del nou model sense perdre l'avantatge que suposa l'ús d'una part d'una xarxa que ja ha sigut entrenada és la modificació del paràmetre corresponent al rati d'aprenentatge, que controla la magnitud de l'actualització dels pesos durant el procés d'optimització. Per tant, es va especificar un rati molt baix (1e-4) per a la part reaprofitada, i un de molt alt per a les capes noves a entrenar en la tasca de classificació. Aquesta combinació de ratis d'aprenentatge proporciona un ràpid aprenentatge en les capes finals de la CNN. L'altre punt important a especificar en és el 'MiniBatchSize', que correspon a la quantitat de mostres que s'analitzaran a la vegada durant el procés d'aprenentatge, on una Epoch compren un cicle complet d'entrenament en tot el conjunt de dades. Llavors, el 'MiniBachSize' especifica els paquets de mostres que en cada Epoch s'analitzaran per separat.

```
netTransfer = trainNetwork(trainingImages,layers,options);

[YPred,scores] = classify(netTransfer,validationImages);
```

Figura 3-33. Codi MATLAB per a l'entrenament i predicció de la CNN (Font: Pròpia)

Una vegada especificats totes les opcions que definiran el període d'aprenentatge arriba el moment de començar amb ell. Primer, s'hi fa una passada amb un gran nombre de Epochs, per visualitzar l'evolució de l'aprenentatge del model, per així després poder escollir aquell nombre d'iteracions que genera un model amb una millor capacitat de generalització front mostres no vistes.

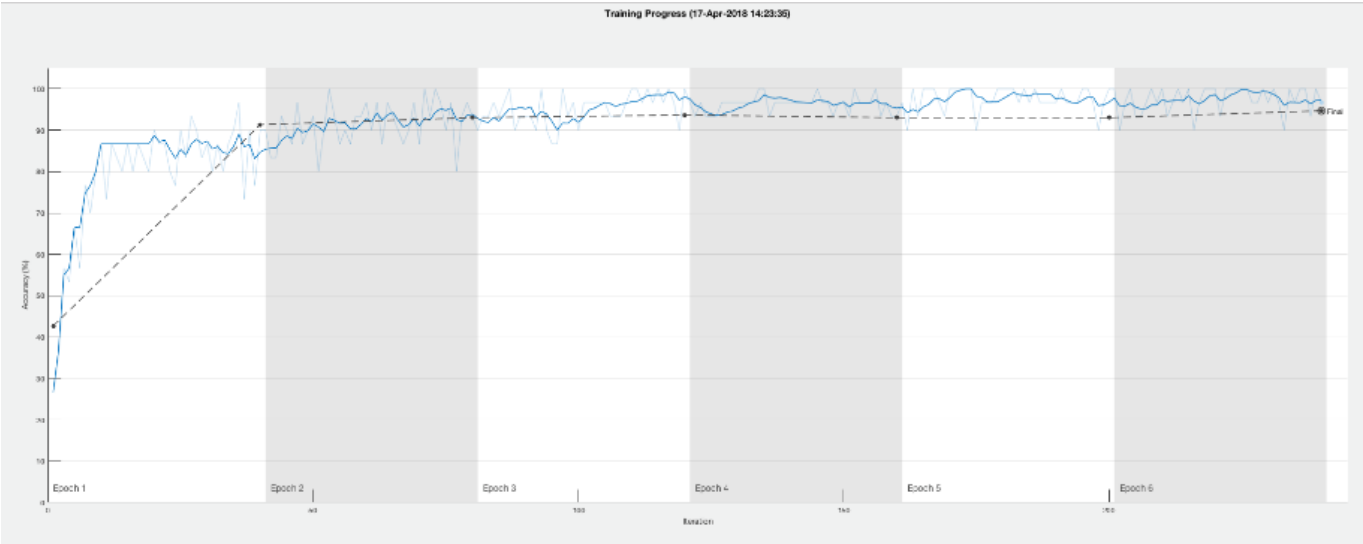


Figura 3-34 Evolució del procés d'entrenament (Font: Pròpia)

En aquest cas, es va observar que més o menys des de la quarta iteració el procés d'entrenament del model ja no implica millores significants. Per tant, es va decidir de tornar a realitzar el procés d'aprenentatge, però parant-lo a la quarta Epoch. Així, s'eviten possibles fenòmens d'Overfitting, ja que una vegada ja no pot continuar aprenent de forma generalitzada, comença a aprendre característiques pròpies de cada mostra, resultant d'això un model no eficaç front mostres mai vistes.

3.5.2. VGG19

En aquest cas també s'ha reaprofitat tot el codi referent a l'adequació de les dades per a fer-les utilitzables amb la xarxa VGG19 explicat a l'apartat d'utilització de CNN com a Extractor de Característiques. La única cosa que s'hi va afegir va ser el codi necessari per convertir les etiquetes de les mostres al format 'Categorical', que és el exigit per les xarxes neuronals per a l'entrenament.

En aquest experiment, la part del model VGG19 entrenat que s'ha utilitzat per fer la transferència dels pesos apresos ha sigut de tot el model menys l'última capa, la de classificació. Aquesta s'ha definit per a distingir entre tres classes i s'ha afegit al darrere de la 'fc2', que correspon a la última 'Fully Connected' conservada del model VGG19.

A diferència del procediment realitzat en l'experiment amb AlexNet, aquí s'ha realitzat el procés d'aprenentatge en dues etapes diferents. La primera, on totes les capes provinents de l'anterior model han tingut els pesos congelats, i per tant durant la optimització sols s'han actualitzat els pesos de la capa de classificació que hem afegit.

```
for layer in model.layers[:17]:
   layer.trainable = False
for layer in model.layers[17:]:
   layer.trainable = True

from keras.optimizers import SGD
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9),
              loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(trainingImages, trainingLabels,
                    validation_data=(testImages, testLabels), epochs=10,  verbose=2)
```

Figura 3-35. Codi Python per a la primera part del mètode (Font: Pròpia)

I una segona part on disposant d'un classificador que comença a saber diferenciar entre classes, es descongelen les últimes capes de convolució i les de 'Fully Connected', i s'especifica un valor baix per al rati d'aprenentatge, amb això s'aconsegueix que com el valor de la funció objectiu ja no és tan dolent, els pesos d'aquestes capes s'actualitzen mínimament per adequar-se completament als atributs del nostre conjunt de dades. Obtenint així un model amb més capacitat de captar els patrons més abstractes de les nostres imatges.

```
base_model = VGG19(weights='imagenet')
model = Model(inputs=base_model.input, outputs=base_model.get_layer('fc2').output)
x = model.output
predictions = Dense(3, activation='softmax')(x)
NouModel = Model(inputs=model.input, outputs=predictions)

for layer in model.layers:
    layer.trainable = False

model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
              metrics=['accuracy'])
history = model.fit(trainingImages, trainingLabels,
                    validation_data=(testImages, testLabels), epochs=10,  verbose=2)
```

Figura 3-36. Codi Python per a la segona part del mètode (Font: Pròpia)

Aquí també es va realitzar un estudi per determinar el nombre d'iteracions òptim per a cada part del procés d'aprenentatge. Obtenint per a la primera part del procés un nombre de set iteracions, com es pot apreciar en la Figura 3-37. I per al segon, un nombre de deu iteracions. La Figura 3-37 correspon a la primera prova per determinar quan comença a haver-hi Overfitting en el procés d'aprenentatge, i a la Figura 3-38, es mostra el procés d'aprenentatge corresponent a l'evolució del període d'aprenentatge definitiu una vegada feta la tria del nombre òptim d'iteracions.

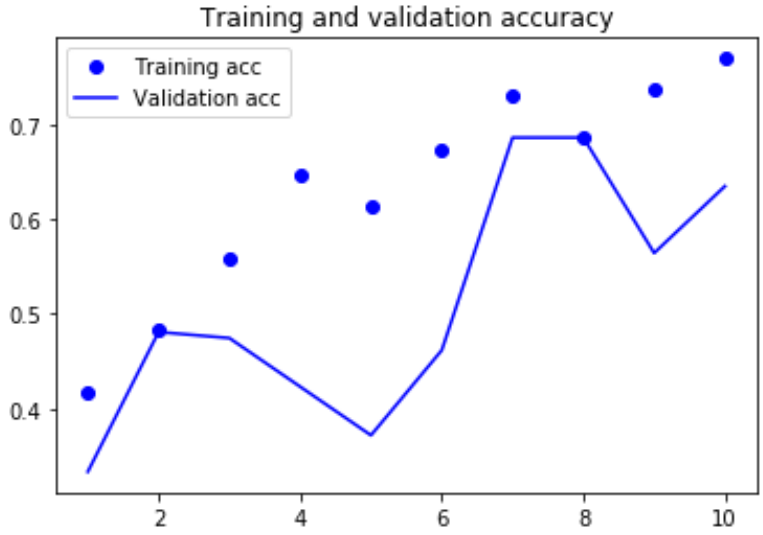


Figura 3-37. Evolució de la primera part del procés d'entrenament (Font: Pròpia)

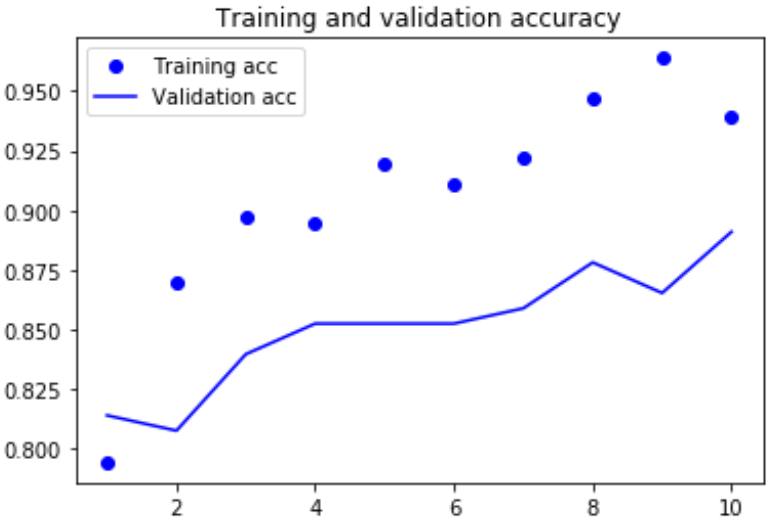


Figura 3-38. Evolució de la segona part del procés d'entrenament (Font: Pròpia)

Si una vegada comprovats els resultats del model creat, aquests són satisfactoris, el següent pas seria el d'entrenar el model que creat amb la totalitat de mostres de que es disposen i desar-lo per tal de procedir a la seva utilització comercial o científica.

3.5.3. InceptionV3

Al igual que ens altres casos, s'han reutilitzat les línies de codi referents a l'adequació de les dades per al seu ús amb la CNN InceptionV3. I com en l'anterior apartat, s'ha tingut que convertir les dades de les etiquetes de les mostres a format 'Categorical'.

En quan a l'elecció de les capes a reutilitzar, el procediment ha sigut diferent al dels casos anteriors. Això, es degut a l'estructura pròpia de la xarxa que s'ha utilitzat, perquè a diferència de les altres, la CNN InceptionV3 no té cap capa del tipus 'Fully Connected' abans de l'última de classificació, sinó que utilitza les capes del bloc de convolució per anar reduint progressivament la dimensionalitat de les dades. Per tant, el procediment seguit el de treure les tres últimes capes de la xarxa i substituir-les per d'altres. És passa de tenir una xarxa on les tres últimes capes estaven constituïdes per layers del tipus 'AveragePooling', 'Flatten' i 'Fully Connected', a tenir-ne una amb 'GlobalAveragePooling2D', 'Fully Connected' i 'Fully Connected' amb l'activació 'softmax'.

De nou, el procés d'aprenentatge ha sigut dividit en dues parts. La primera, on s'han congelat les capes extretes de la xarxa InceptionV3 entrenada, i que transfereixen el coneixement adquirit a través dels pesos de les diferents capes, entrenant així sols la nova part que hem afegit nosaltres. Una altra manera de veureu seria que la part reaprofitada treballa com a extractor de característiques per així entrenar el classificador.

```
base_model = InceptionV3(weights='imagenet', include_top=False)
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(3, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)

for layer in base_model.layers:
    layer.trainable = False

model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
              metrics=['accuracy'])
history = model.fit(trainingImages, trainingLabels, epochs=10,  verbose=2)
```

Figura 3-39. Codi Python per definir les noves capes (Font: Pròpia)

Com a l'experiment anterior, s'ha realitzat un estudi per tal de trobar el nombre d'iteracions òptim del procés d'aprenentatge. Seleccionant un nombre d'iteracions molt alt i mirant com evoluciona per trobar on comença a deixar d'aprendre patrons generals i així evitar el Overfitting. Per aquest cas, es va utilitzar un nombre inicial de deu iteracions, que després de l'estudi dels resultats vàrem reduir a set.

A la segona part del procés d'aprenentatge, és a dir, una vegada entrenat el classificador per a distingir entre les diferents classes, es varen descongelar els dos últims blocs Inception (estructura de capes de convolució) per a procedir al seu afinament per a es característiques pròpies de les nostres imatges. Per això, s'ha especificat un valor molt petit per al rati d'aprenentatge, així, s'aprofita tot el coneixement del model reutilitzat i es perfilen els detectors per a les característiques dels nous tipus de patrons.

```
for layer in model.layers[:249]:
   layer.trainable = False
for layer in model.layers[249:]:
   layer.trainable = True

from keras.optimizers import SGD
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9),
              loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(trainingImages, trainingLabels,
                    validation_data=(testImages, testLabels), epochs=4,  verbose=2)

resultats=model.predict(testImages)
```

Figura 3-40. Codi Python per especificar paràmetres (Font: Pròpia)

En aquest cas, després de realitzar l'estudi del nombre d'iteracions idoni per a aquesta part del procés d'aprenentatge, s'ha trobat que amb quatre iteracions de tot el procés ja s'assoleix el màxim nivell de generalització possible.

Una vegada trobades les condicions òptimes per a l'obtenció del millor model possible, l'últim pas abans de la seva utilització o comercialització seria el d'entrenar el model tal com s'ha descrit amb la totalitat de les mostres, i desar el model obtingut en un format que faciliti la seva utilització.

3.6. Disseny d'una CNN a mida

En aquest apartat es procedirà a l'exposició dels passos seguits i les consideracions tingudes en compte durant el procés de disseny d'una CNN per solucionar satisfactòriament el problema que es té en consideració en aquest projecte.

El procediment de treball recomanat per a quan es pretén dissenyar una xarxa neuronal des de zero consisteix en començar amb un disseny el més simple possible, el qual suposadament sofrirà Underfitting, i anar paulatinament augmentant la seva complexitat fins trobar que al entrenar-lo amb les nostres dades es produeix Overfitting, i llavors reduir una mica la seva complexitat fins observar que abans de produir-se Overfitting, el model ha aprés a diferenciar entre les diferents classes satisfactòriament. En aquest moment s'haurà trobat un bon disseny per a la xarxa i el nombre d'iteracions òptim segons el tipus d'estructura escollit i els atributs del conjunt de dades.

A continuació exposarem algunes de les consideracions inicials que es varen prendre abans de començar a pensar quina estructura s'utilitzaria. La primera va ser realitzar el canvi de imatge de tres canals (RGB) a altres d'un sol canal (Gray). Durant els altres experiments realitzats s'han utilitzat imatges en format RGB, perquè les xarxes utilitzades varen ser confeccionades i entrenades per imatges a color, format que també es correspon amb l'original de es imatges del nostre data set, però que una vegada després de realitzar el preprocessat, consistien en imatges en escala de gris repetides en els tres canals. Així que seleccionant un sol dels tres canals s'hi obté la mateixa informació de la imatge, i això suposa reduir dràsticament la despesa computacional per entrenar i utilitzar el model. La segona consideració presa va ser referent a la mida que devien tenir les imatges que utilitzaríem com entrada per a la xarxa. Com s'ha anat veient al llarg dels diferents experiments, les mides de les imatges sempre han seguit el requisit de tenir el mateix nombre de píxels d'alçada que d'amplada, i la magnitud d'aquests no sol sobrepassar els 300*300 píxels, per tant, es va decidir que s'utilitzarien les imatge preparades per a treballar amb AlexNet, que tenien una mida de 227*227*3, però com ja s'ha explicat, sols s'agafaria un dels tres canals, quedant finalment així: 227*227*1.

Un altra modificació a tenir en compte es la partició del conjunt de dades per a la realització de l'experiment. Com sempre, es necessari de la disposició d'un gran volum de mostres per a l'entrenament del model i d'un altre conjunt més petit per a comprovar el funcionament final del conjunt i la seva efectivitat en la tasca de classificació. Però en aquest cas també ha resultat necessari un tercer conjunt, que anomenarem de validació del disseny, per ajustar l'estructura i el disseny del model, aquests conjunt s'utilitzarà per comprovar com afecten les diverses modificacions. L'ús d'aquest tercer conjunt és molt important, ja que si modifiquem el disseny del model en referència als resultats que obté amb el conjunt de validació, es corre el risc de desenvolupar un model a mida per a aquestes mostres de validació, i que no sigui capaç de generalitzar per a altres mostres. Per tant, es disposa de tres conjunts de dades, el d'entrenament amb 386 mostres de cada classe, el de validació del disseny amb 20 mostres per classe, i el de validació o test amb 100 mostres per classe.

En quant a la funció objectiu, com es tracta d'un problema de classificació amb més de dos classes d'imatges, es va utilitzar 'categorical_crossentropy', i per a l'optimitzador, Descens Estocàstic de Gradients.

El disseny inicial va ser el de la Figura 3-41, on com es pot observar es van fer els dos blocs típics de les Xarxes Neuronals de Convolució, el bloc de convolució i el bloc final de capes 'Fully Connected' amb el classificador.

```
def baseline_model():
    model = models.Sequential()
    model.add(layers.Conv2D(32, (3, 3), activation='relu',
                            input_shape=(227, 227, 1)))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.Flatten())
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(3, activation='softmax'))
    model.compile(optimizer=SGD, loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

Figura 3-41. Codi Python per definir la primera versió de la CNN (Font: Pròpia)

El bloc de convolució s'hi trobava format per tres capes de convolució amb finestres d'una mida de 3*3. Les dues primeres s'hi trobaven seguides per capes de 'MaxPooling' per tal de realitzar reduccions de dimensionalitat, i quedar-se sols amb els patrons clarament trobats.

El segon bloc anava a continuació d'una capa de convolució, per tant, la primera capa de totes és del tipus 'Flatten', així s'aconsegueix posar les dades en un format en que les 'Fully Connected' hi poden treballar. La següent capa la constituïa una de 'Fully Connected' normal, la funció de la qual era unir tots els atributs locals captats per el bloc de convolució i així fer una representació global de la mostra. I per últim, una altra 'Fully Connected' però amb l'activació 'softmax' per a poder treballar com a classificador. L'activació escollida per a totes les altres capes ha sigut la 'relu'.

Després de sotmetre aquest model a un procés d'aprenentatge, hi vam trobar que no era capaç d'aprendre a diferenciar entre les diferents classes, és a dir, ens trobàvem davant un clar cas d'Underfitting. Per tant, els passos a seguir consistien en augmentar la complexitat de la xarxa.

Primer es va optar per augmentar el nombre de pesos de la penúltima capa, per veure si al realitzar una reducció tan gran de la dimensionalitat dels atributs per a la classificació el que realment estàvem fent era perdre informació vital per a la tasca. Així que es va augmentar de 64 pesos a 400.

Al tornar a provar que passava si entrenàvem el model, observarem que amb el canvi realitzat tampoc no s'havia aconseguit resoldre el problema. Per tant, es va decidir modificar el bloc de convolució, ja que possiblement el que passava era que era tan simple que no era capaç d'extraure patrons i atributs clars de les imatges.

Per a la segona prova s'hi van afegir dues capes més de convolució, i tres de 'MaxPooling'. Quedant sempre una de 'MaxPooling' després d'una de 'Conv2D'. Tot i que en el primer intent el canvi no va suposar cap millora aparent, es va decidir de mantenir el canvi realitzat, ja que ben pensat, així tenia més sentit que com estava.

En aquest moment, es va observar que durant el procés d'aprenentatge el model ja deixava de classificar les imatges de forma aleatòria i que era capaç d'aprendre certs patrons identificatius, però que els resultats eren encara més que desitjables. Així que es va decidir d'augmentar el nombre de capes de convolució de les primeres etapes per veure si així el model esdevenia un millor detector i identificador de patrons més abstractes.

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(227, 227, 1)))
model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(600, activation='relu'))
model.add(layers.Dense(3, activation='softmax'))
model.compile(optimizer=SGD, loss='categorical_crossentropy',
                  metrics=['accuracy'])
```

Figura 3-42. Codi Python per definir la versió final de la CNN (Font: Pròpia)

Després de diverses probes es va arribar al disseny de la base de convolució que trobem a la Figura 3-41, on agafant com inspiració de les CNN amb que s'ha treballat durant el projecte, es comença primer amb diverses etapes de convolució seguides, per anar reduint el nombre de capes de convolució d'aquestes etapes com més endins de la xarxa.

Per últim, es va decidir de tornar a canviar el nombre de pesos de la penúltima capa, que fins al moment havia estat de 400, i es va veure que primer amb 1000 i després amb 800, el model experimentava Overfitting molt sobtadament, per tant, es va continuar reduint el nombre de pesos fins arribar a 600, on sorprenentment es va observar com el model experimentava una molt bona fase d'aprenentatge, generalitzant correctament per a les imatges de validació del disseny. Fent que d'aquesta, la última modificació de la CNN, on el resultat final és el que es veu a la Figura 3-42.

4. Resultats

Els resultats del projecte es troben descrits i analitzats en relació a la metodologia i els passos proposats en el capítol 3. Segons el nostre propi punt de vista, les solucions trobades al problema de classificació mitjançant la utilització de les Xarxes Neuronals de Convolució han sigut molt satisfactoris en la tasca de reconeixement dels Plans Biomètrics Fetals Standard.

4.1. Organització dels experiments

Per a la classificació dels plans Biomètrics Fetals representatius de les diferents mesures tingudes en compte, la següent configuració experimental va ser la seguida:

- **Experiment 1:** Preprocessat del conjunt d'imatges. La metodologia seguida en aquesta etapa s'hi troba basada en el desenvolupament successiu de diferents passos per netejar el conjunt de dades d'anotacions de text i de color irrellevants, resaltant les estructures anatòmiques importants de les imatges ecogràfiques. Amb l'objectiu de guanyar més uniformitat, s'hi van aplicar tècniques per a realçar la intensitat i reduir el soroll de la imatge.

- **Experiment 2:** Extracció de característiques de les imatges mitjançant part d'una CNN prèviament entrenada, i entrenament d'un SVM amb aquestes característiques extretes per a la seva utilització com a classificador. Després d'extraure tots els artefactes de text i color presents a les imatges, es van obtindré les característiques de les imatges via una CNN i es va avaluar quin tipus de classificador entre els SVM és el més apropiat per al nostre problema. Finalment, vàrem trobar que el classificador més adient era utilitzar un SVM amb kernel polinòmic de segon grau per al reconeixement de les característiques de les imatges ecogràfiques, i que per a l'extracció de característiques, la CNN més efectiva resulta ser AlexNet.

- **Experiment 3:** Creació d'una nova Xarxa Neuronal de Convolució transferint part d'una altra entrenada prèviament, aprofitant així allò que ha après durant l'entrenament a un problema més complex. Entrenament d'aquest nou model amb les dades del nostre problema, i avaluació dels resultats del nou classificador després de l'entrenament. Es trobà que per aquest experiment, la CNN que s'ha utilitzat per a la transferència de coneixement més efectiva és AlexNet.

- **Experiment 4:** Creació d'una nova Xarxa Neuronal de Convolució des de zero. Avaluació del model després de cada modificació per tal de trobar aquelles que portin al model òptim. Avaluació del model final i comparació amb els resultats dels altres experiments.

4.2. Resultats del preprocessat del conjunt d'imatges

En aquesta secció es presentaran els resultats obtinguts per a cadascuna de les classes d'imatges de que disposem després d'aplicar les tècniques de preprocessat descrites. A la Figura 4-1 s'hi observa un exemple de varies mostres de la classe d'imatges BPD. Com es pot veure, amb el preprocessat s'ha aconseguit que l'estructura anatòmica del cap hagi quedat realçada i apropiadament visible en aquells casos on fins i tot era difícil de visualitzar manualment.

Per a aquesta classe, trobem que l'automatització del preprocessat ha tingut un efecte satisfactori, ja que s'han aconseguit extreure les marques de color i reduir significativament tant les parts de text com el soroll.

Figura 4-1. Resultats del preprocessat per a les imatges BPDt (Font: Pròpia)

En la Figura 4-2 s'hi pot veure una mostra dels resultats obtinguts al aplicar el procés de preprocessat a les imatges respectives a la classe CRL. S'ha observat que els resultats obtinguts són similars als de la classe anterior, és a dir, s'ha aconseguit l'eliminació de les marques de color i una reducció considerable del soroll i del text, però realçant la intensitat d'aquelles zones que representen estructures anatòmiques importants del fetus.

Figura 4-2. Resultat del preprocessat per a imatges CRL (Font: Pròpia)

Per últim, per a les imatges de la classe NT el mètode utilitzat també pareix resultar eficaç. Es poden observar un conjunt de mostres d'imatges d'aquest tipus en la Figura 4-3. La vista sagital de l'anatomia del cap del fetus mostra com s'ha realçat la intensitat en aquells punts on referents a l'estructura del cap i del plec nucal, que eren les característiques més transcendentals a mantenir i realçar. A les mostres de la Figura 4-3 també observem com anotacions i les marques de color han sigut correctament eliminades de les regions del fetus.

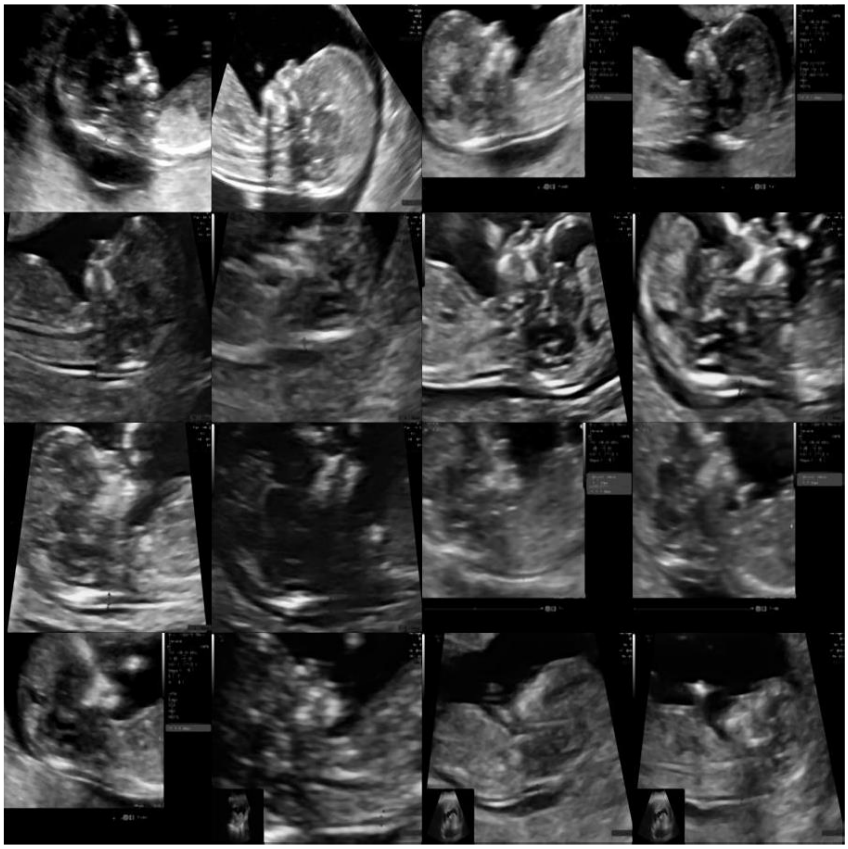


Figura 4-3. Resultat del preprocessat per a imatges NT (Font: Pròpia)

Com a conclusió, els resultats que s'han obtingut d'aplicar el mètode de preprocessat descrit a les imatges, mostren una clara millora de la qualitat de les imatges, fet que es traduirà en una optimització dels resultats que s'obtindran en les següents etapes tant d'extracció de característiques, com de classificació del nostre conjunt d'imatges ecogràfiques .

4.3. Resultats de la classificació

4.3.1. CNN com a Extractor de Característiques

4.3.1.1 AlexNet

L'execució de la CNN AlexNet sense l'última capa (de classificació) amb el conjunt d'imatges ecogràfiques va generar una taula de dades on per cada fila, les quals representen les mostres, hi havia 4096 característiques. El nombre de característiques per mostra es troba definit pel nombre de neurones (pesos) de la última capa, que en aquest cas és de 4096.

Un total de 506 imatges de cada classe va propiciar que obtinguérem una taula de dades resultant amb 1518 mostres amb les respectives característiques de cadascuna, i una altra taula amb 1518 mostres i les etiquetes referents a la classe a la que pertany cada imatge. Per a obtenir uns resultats més representatius es va entrenar el SVM utilitzant el protocol de validació "k-fold cross-validation" amb K=5, conegut com Validació Creuada. Aquest protocol ha sigut el que s'ha fet servir per a tots els experiments del present projecte.

Un dels passos que es va tenir que realitzar en els altres experiments va ser l'estudi de la idoneïtat de una normalització de les imatges, però en el cas de la CNN AlexNet, perquè a les indicacions de la web de MATLAB s'indica que no és necessari cap tipus de normalització.

La matriu de confusió representada a la Taula 4.1 mostra els resultats de la classificació dels diferents conjunts d'imatges de validació del k-fold amb el SVM resultant. Els quadrants més significatius de la matriu de confusió són els que es troben de color verd, ja que aquests representen el rati d'imatges de cada classe que s'han classificat correctament com a representació dels Plans Biomètrics Fetals. Com es pot observar, tot i que els resultats es poden considerar molt bons, hi trobem que el classificador ha tingut alguns problemes per reconèixer diverses mostres, causant que un 5.8% de les mostres corresponents al tipus BPD hagin sigut classificades com a CRL, i un altre 5.8% com a NT, o que un 5.8% de les mostres de la classe CRL hagin sigut identificades com a NT.

		Classe predita				
		BPD	CRL	NT	Sensitivitat (TPR)	Especificitat (TNR)
Classe real	BPD	88.5%	5.8%	5.8%	88.5%	98.05%
	CRL	2%	92.3%	5.8%	92.3%	96.15%
	NT	2%	2%	96%	96%	94.3%

Taula 4-1. Resultat de la classificació amb Validació Creuada (Font: Pròpia)

4.3.1.2 VGG19

L'execució de la CNN VGG19 sense l'última capa (de classificació) amb el conjunt d'imatges ecogràfiques va generar una taula de dades on per cada fila, les quals representen les mostres del conjunt de dades, hi havia 4096 característiques. El nombre de característiques per mostra es troba definit pel nombre de neurones (pesos) de la última capa, que en aquest cas és de 4096.

Un total de 506 imatges de cada classe va propiciar que obtinguérem una taula de dades resultant amb 1518 mostres amb les respectives característiques de cadascuna, i una altra taula amb 1518 mostres i les etiquetes referents a la classe a la que pertany cada imatge.

Mitjançant la tècnica explicada a l'anterior capítol, es va trobar que un dels classificadors amb que millor resultats s'obtenien era un SVM amb kernel polinòmic de grau 2 i amb C=100, i que a més es tracta del més senzill de entre tots aquells que donaven el mateix resultat. Aquesta selecció es va fer seguint el criteri comú en Machine Learning de quan diversos models donen igual resultat per al mateix problema sempre quedar-se amb el més senzill, ja que això evita possibles fenòmens d'Overfitting.

També es va realitzar un estudi per trobar la idoneïtat d'aplicar un procés de normalització de les imatges, però que en el cas de l'experiment que estem analitzant ara els resultats van ser els següents:

	Normalització de la escala per a cada canal (/255)	Normalització per estandardització	Sense normalitzar
Resultats	74%	77.66%	87.66%

Taula 4-2. Comparació dels resultat segons la normalització(Font: Pròpia)

Com es pot veure a la Taula 4-2, una vegada realitzat aquest petit estudi, es va trobar que el més idoni per treballar amb la xarxa VGG19 era la utilització de les imatges sense haver sigut sotmeses a cap normalització, ja que per aquest cas s'aconsegueix una classificació eficaç en el 87.66% dels casos, eficàcia un 10% major que en les altres opcions.

La matriu de confusió de la Taula 4-3 mostra els resultats dels processos de classificació del SVM amb els conjunts d'imatges de validació. Com es pot observar, tot i que els resultats es poden considerar bons, hi trobem que el classificador ha tingut més problemes per reconèixer diverses mostres que en l'experiment anterior, causant que un 10.6% de les mostres corresponents al tipus BPD hagin sigut classificades com a CRL o que un 10.9% de les mostres de la classe CRL hagin sigut identificades com a NT.

		Classe predita				
		BPD	CRL	NT	Sensitivitat (TPR)	Especificitat (TNR)
Classe real	BPD	86%	10.6%	4.4%	86%	97%
	CRL	3.8%	85.3%	10.9%	85.3%	92.2%
	NT	2.3%	6%	91.7%	91.7%	92.9%

Taula 4-3. Resultat de la classificació amb Validació Creuada (Font: Pròpia)

A més a més, la Sensitivitat, també coneguda com True Positive Rate(TPR), i la Especificitat, coneguda com True Negative Rate (TNR), aporten més informació de l'escenari resultant i ens mostren més diferències entre els casos correctes i els incorrectes, donant una idea més clara de perquè el SVM ha fallat en la identificació i classificació d'algunes de les mostres. En aquest context, les diferències entre els valors d'especificitat i sensibilitat per a les diferents classes no varien en gran mesura, per la qual

cosa no es fàcil de trobar una manera de amb el conjunt de dades disponible augmentar més aquests valors.

4.3.1.1 InceptionV3

L'execució de la CNN InceptionV3 sense les dues últimes capes (una del tipus 'Flatten' i la de classificació) amb el conjunt d'imatges ecogràfiques va generar una taula de dades on per cada fila, les quals representen les mostres del conjunt de dades, hi havia 2048 característiques. El nombre de característiques per mostra es troba definit pel nombre de finestres de 'AveragePooling' aplicades a les dades d'entrada de la última capa.

En quant a l'estudi d'idoneïtat de la normalització del conjunt de dades els resultats són els següents:

	Normalització de la escala per a cada canal (/255)	Normalització per estandardització	Sense normalitzar
Resultats	92.67%	82.33%	36%

Taula 4-4. Comparació dels **r**esultat segons la normalització(Font: Pròpia)

Com es pot veure en la Taula 4-4, al realitzar aquest petit experiment, trobem que en el cas específic de la xarxa InceptionV3 el més adient és aplicar una normalització dels valors per a cada canal, fent que tots els valors dels píxels estiguin en un rang de [0, 1]. El resultat d'aplicar aquest tipus de normalització al conjunt de dades resulta en una eficàcia en el procés de classificació d'un 92.67%, valor infinitament superior al cas sense normalització i un 10% superior al de la normalització estàndard.

La matriu de confusió de la Taula 4-5 mostra els resultats dels processos de classificació del SVM amb els conjunts d'imatges de validació. Com es pot observar, tot i que els resultats es poden considerar molt bons, de fet millors que en l'experiment on s'ha utilitzat la CNN VGG19 com a Feature Extractor, i amb un resultat pràcticament igual al del cas on s'ha utilitzat AlexNet.

		Classe predita				
		BPD	CRL	NT	Sensitivitat (TPR)	Especificitat (TNR)
Classe real	BPD	94.5%	5%	0.5%	94.5%	98.4%
	CRL	2.1%	88.1%	9.8%	88.1%	95.7%
	NT	1.1%	3.7%	95.2%	95.2%	94.9%

Taula 4-5. Resultat de la classificació amb Validació Creuada (Font: Pròpia)

Al igual que ens casos anteriors, les diferències entre els valors d'especificitat i sensibilitat per a les diferents classes no varien en una mesura suficientment gran com per trobar una manera de amb el conjunt de dades disponible augmentar més aquests valors.

Amb la finalitat de trobar una manera d'aconseguir una millor eficàcia del classificador es va provar de realitzar aquests mateixos tres experiments amb un conjunt de dades reduït per tal de veure com afecta el volum de dades disponible per a l'entrenament del classificador, però els resultats varen mostrar que les diferències no eren significatives, és a dir, augmentant el volum de dades d'entrenament s'aconseguia incrementar l'eficàcia del classificador en unes dècimes.

4.3.2. Classificació amb CNN per Transferència de l'Aprenentatge

4.3.2.1 AlexNet

L'execució de la nova CNN creada a partir de la transferència dels pesos apresos per AlexNet (mitjançant el conjunt d'imatges ImageNet) sense l'última capa (de classificació), que es substituïda per una de nova capa a mida per a l'entrenament amb el conjunt d'imatges ecogràfiques, va generar una un nou model capaç de discriminar entre els diferents plans d'ecografia prenatal tinguts en compte per al problema.

Els resultats que hi troben en la Taula 4-6 són els referents al procés d'aprenentatge al que ha sigut sotmesa aquesta CNN després de trobar el nombre ideal de iteracions del procés, que va resultar ser de 4 Epochs.

		Classe predita				
		BPD	CRL	NT	Sensitivitat (TPR)	Especificitat (TNR)
Classe real	BPD	95%	4%	1%	95%	98.5%
	CRL	1%	97%	2%	97%	95%
	NT	2%	6%	92%	92%	98.5%

Taula 4-6. Resultat de la classificació amb Validació Creuada (Font: Pròpia)

Per tal de trobar uns resultats realment representatius, i que no s'hi trobaren sotmesos a l'aleatorietat en que pot desencadenar l'elecció de la partició del conjunt de dades per a entrenament i per a validació, es va utilitzar el mètode de Validació Creuada amb (K=5). Resultant en cinc processos d'entrenament i el càlcul de la mitja aritmètica dels resultats. On per a cada una de les classes, un 80% de les imatges fan d'entrenament (unes 405), i un 20% són utilitzades per a validació (unes 101).

La matriu de confusió de la Taula 4-6 mostra els resultats obtinguts després d'aplicar el procés explicat anteriorment, i que permet analitzar l'actuació del model i la tècnica utilitzada per a la resolució del problema. Trobem que el model té una eficàcia en la tasca de classificació del 94.66%, que resulta la més elevada entre les trobades fins al moment. Però tot i això, s'observa que encara que molt més

reduïts, continua tenint alguns problemes al classificar mostres representatives de les classes BPD i NT com a CRL.

4.3.2.2 VGG19

En aquest apartat es tractarà d'analitzar els resultats obtinguts en l'experiment corresponent a crear una nova xarxa CNN a partir de transferir els pesos apresos per la CNN VGG19 amb el conjunt de dades ImageNet i afegir una nova capa de classificació per adequar-la al nostre problema. Com ja s'ha explicat en el capítol anterior, per aquest cas, el procés d'aprenentatge es troba dividit en dues etapes, una on totes les capes procedents de la CNN VGG19 s'hi troben congelades, i per tant sols s'entrena la capa de classificació durant 7 Epochs (Iteracions de tot el procés), i una segona, on es descongelen part de les últimes capes i s'entrenen per adequar-se als atributs propis de les nostres imatges durant 10 Epochs.

Per tal de trobar uns resultats que no estiguessin influenciats per l'aleatorietat en l'elecció de les particions del conjunt de dades per a l'entrenament i la validació, és va decidir d'aplicar una Validació Creuada amb K=5.

La matriu de confusió de la Taula 4-7 mostra els resultats d'aplicar el mètode de la Validació Creuada. L'eficàcia del model ha resultat ser d'un 89.1%, la qual cosa significa que quasi una de cada deu mostres no és classificada com a representació del Pla Biomètric Fetal del que realment es correspon, encara que més baixa que en el cas anterior, es correspon amb un resultat molt bo.

		Classe predita				
		BPD	CRL	NT	Sensitivitat (TPR)	Especificitat (TNR)
Classe real	BPD	90.3%	7.1%	2.6%	90.3%	96.4%
	CRL	4.2%	85.3%	10.5%	85.3%	93.9%
	NT	3.1%	5.2%	91.7%	91.7%	93.5%

Taula 4-7. Resultat de la classificació amb Validació Creuada (Font: Pròpia)

En quant als valors de sensitivitat i especificitat que s'han obtingut, s'hi observa com al igual que en tots els altres experiments que s'han realitzat, el model continua tenint problemes sobretot a l'hora de diferenciar entre les classes CRL i NT, cosa que sembla raonable, ja que aquestes dues classes presenten moltes similituds, tant de les estructures anatòmiques que hi surten, com de la disposició en que s'hi troba el fetus a l'hora de realitzar la captura de la imatge. Per tant, podem suposar que al detectar moltes de les estructures pròpies d'aquestes classes en la classe que realment no li pertoca, el model es confon i acaba classificant aquestes mostres on no és.

4.3.2.3 InceptionV3

En quant als resultats obtinguts en aquest experiment, on s'ha realitzat la transferència de pesos apresos des de la CNN InceptionV3, i també s'ha utilitzat la tècnica de Validació Creuada amb K=5, són els que trobem a la Taula 4-8. Aquests, són els més baixos de entre tots els obtinguts fins al moment.

		Classe predita				
		BPD	CRL	NT	Sensitivitat (TPR)	Especificitat (TNR)
Classe real	BPD	79.6%	11.7%	8.7%	79.6%	90.4%
	CRL	11.4%	72.2%	16.4%	72.2%	89.2%
	NT	7.9%	9.9%	82.2%	82.2%	87.5%

Taula 4-8. Resultat de la classificació amb Validació Creuada (Font: Pròpia)

La diferència més significativa entre el model resultant d'aquest experiment i el resultant de tots els anteriors és principalment la seva llargada. Com hem vist al capítol anterior, el conjunt de capes de convolució de la CNN InceptionV3 és molt més gran que en la VGG19 i en AlexNet, la qual cosa li permet arribar a un nivell d'abstracció major a l'hora de la detecció de patrons. El que potser ocorre és o que s'arriba a un nivell d'abstracció superior al que presenten les estructures de les nostres imatges, o que les capes de classificació no estiguin resultant capaces de reconèixer quins atributs són representatius de cada tipus de classe.

4.3.3. CNN a mida

En aquest apartat hi presentem els resultats finals de la CNN que s'ha creat des de zero. Tots els resultats previs varen servir per a tenir un criteri clar a l'hora de modificar l'estructura de la CNN. Com que tot el procés de disseny i situacions tingudes en compte s'han detallat en el capítol anterior, aquí sols presentarem els resultats referents a l'estructura final del model.

		Classe predita			Sensitivitat (TPR)	Especificitat (TNR)
		BPD	CRL	NT		
Classe real	BPD	90.3%	5.7%	4%	90.3%	93%
	CRL	10%	87.6%	2.4%	87.6%	93.1%
	NT	4%	8.2%	87.8%	87.8%	96.8%

Taula 4-9. Resultat de la classificació amb Validació Creuada (Font: Pròpia)

A la Taula 4-9 s'hi presenta la matriu de confusió resultant dels procés de Validació Creuada aplicat al model, on per a 1518 mostres (506 per classe) s'ha utilitzat un K=5 per al mètode de validació seleccionat. Tot i que els resultats obtinguts són molt bons, es troben per baix de l'eficàcia trobada amb l'experiment on s'han utilitzat els pesos d'AlexNet per a la transferència de l'aprenentatge. Aquesta diferència en els resultats pot ser deguda a com que l'aprenentatge de la CNN comença des de zero, i el nombre d'imatges del nostre conjunt és petit en comparació al recomanat per a utilitzar tècniques de Deep Learning, la disponibilitat d'unes capes ja entrenades en la detecció de patrons alleuja molt el procés d'optimització, permetent una major convergència cap al mínim global. De nou, creiem que la solució per a millorar l'efectivitat en la classificació per a aquest model és redueix bàsicament en l'augment del conjunt de dades.

4.3.4. Visió general

En aquest apartat ens disposem a analitzar el conjunt total de resultats obtinguts en els diferents experiments realitzats. Com ja sabem, durant l'elaboració del projecte s'han seguit tres metodologies per tal de trobar una solució al problema. Els tres mètodes consistien en la utilització de CNN en alguna part del procés, per tant, els diferents experiments realitzats s'han dividit segons el mode d'ús de les CNN, i la CNN utilitzada en cada cas.

	AlexNet	VGG19	InceptionV3	Vicent
Extractor de Característiques + SVM	92.3%	87.7%	92.6%	
Transferència de l'aprenentatge	94.67%	89.1%	78%	
Disseny d'una CNN a mida				88.7%

Taula 4-10. Comparació dels resultats de tots els experiments(Font: Pròpia)

Pel que fa als experiments on s'ha utilitzat una CNN coma Extractor de Característiques, una observació detinguda dels resultats porta a la conclusió final de com que tant quan s'ha utilitzat VGG19, com InceptionV3, el tipus de classificador i els seus paràmetres han sigut completament iguals. Per tant, la diferència en els resultats ve deguda a que per a les estructures anatòmiques de les imatges del nostre problema, la CNN InceptionV3 actua amb major eficàcia extraient els atributs representatius de cada tipus de pla ecogràfic, és a dir, els patrons apresos per aquesta CNN al ser entrenada amb el conjunt d'imatges ImageNet constitueixen un millor identificador dels patrons presents a les nostres imatges.

Com ja hem comentat abans, el criteri comú en els problemes on s'utilitza Machine Learning es basa en quan es doni el cas que diferents models proporcionen els mateixos resultats, sempre s'ha de elegir aquell més senzill, fugint de la complexitat, ja que pot esdevenir en fenòmens d'Overfitting. Per tant, dels tres experiments realitzats utilitzant una CNN com a Feature Extractor, ens quedem amb el que utilitza AlexNet, ja que de les tres CNN utilitzades és la que té una estructura menys complexa, la qual cosa també suposa tindre un cost computacional significativament més baix.

Els resultats obtinguts en el disseny final de la CNN creada a mida per al problema, encara que no siguin els més han sigut superiors als esperats. Ja que un volum de dades com el que disposàvem és petit en relació als utilitzats normalment per a entrenar una CNN, però aquest bon resultat mostra que el procés de disseny de la xarxa ha sigut el correcte.

En canvi, als experiments on les CNN s'han utilitzat per transferir els pesos apresos durant el seu entrenament amb ImageNet a la nova CNN, els resultats obtinguts presenten una variació molt major, amb la qual cosa, l'elecció del millor model resulta més senzilla. En aquest cas, trobem que utilitzant AlexNet el model acaba amb una eficàcia del 94.67%, aquest per tant serà l'opció escollida d'entre els experiments on s'ha realitzat transferència de l'aprenentatge.

La millor solució possible per a augmentar considerablement el resultat passa per augmentar el nombre mostres del nostre conjunt de dades. Ja que en primer lloc sols disposàvem de 172 mostres per a cada classe, i així com els resultats on s'utilitzava una CNN com a Extractor de Característiques i després un SVM com a classificador no variaven significativament, aquí el resultat es troba íntimament lligat amb el volum del conjunt de dades. La diferència de resultats entre aplicar aquest mètode amb la primera base de dades que vàrem tenir i la final, és molt gran, ja que es va passar de tenir una eficàcia del 87% a una de 94.66%.

La conclusió que trobem per tant, és que aquest mètode és el més adient per a resoldre el nostre problema de classificació. Per tant, una vegada trobat quin mètode és el més adient, l'últim pas abans de procedir a la seva utilització ja com a producte acabat, seria l'entrenament del model amb tot el conjunt de dades disponible i exportar-lo a un format que permeti un accés fàcil sempre que es vulgui fer servir.

5. Anàlisi de l'impacte ambiental

L'anàlisi de l'impacte ambiental consisteix en l'avaluació de les conseqüències ambientals esdevingudes per l'ús de les pràctiques i/o tecnologies desenvolupades en un projecte. Per tal de trobar aquestes conseqüències i el seu grau de severitat, es realitza un estudi tècnic, l'objectiu del qual és identificar aquelles pràctiques que porten associades les conseqüències, i així una vegada identificat el problema, tractar d'eliminar o compensar els impactes negatius derivats del projecte. El que és clar és que tots els avanços o accions produiran algun tipus d'impacte ambiental, per tant, mitjançant l'anàlisi de l'impacte ambiental es tracta de determinar el grau d'impacte que es pot considerar acceptable, i reduir l'existent fins assolir aquest nivell.

En quant a aquest projecte, el conjunt de tècniques i les eines utilitzades per a la classificació de Plans Biomètrics Fetals han sigut desenvolupades a través d'algoritmes informàtics utilitzant un ordinador amb el software adequat (Python i MATLAB). Per tant, l'impacte ambiental associat a aquest projecte es redueix a la utilització de l'energia elèctrica com a font d'alimentació de l'ordinador.

Addicionalment, degut a la necessitat de desplaçament amb l'objectiu de recollir les imatges del conjunt de dades, acudir a les reunions de treball del laboratori i al laboratori per fer ús dels ordinadors i reunions de control del correcte desenvolupament del projecte, s'ha fet un gran ús del transport públic de l'Àrea Metropolitana de Barcelona, que té com a impacte ambiental l'ús de combustible.

Conclusions

La classificació automàtica d'imatges, tal i com s'ha pogut observar al llarg del treball, és una disciplina complexa, però amb una gran potència a l'hora de treballar amb grans quantitats de dades, essent els models resultants, eines capaces d'assolir uns millors nivells de rendiment quan a major quantitat de dades haja sigut exposat. Esdevenint un camp molt important en el futur de la tecnologia.

Cal recordar que aquest projecte es troba relacionat amb el Treball de Fi de Grau de Safae Bendali [19]. L'objectiu principal consisteix en ampliar l'estudi de mètodes de classificació per tal de trobar un suficientment eficaç com per constituir una eina amb que automatitzar aquest procés, reduint la càrrega dels professionals sanitaris, i donant la possibilitat de que disposen del seu temps i esforços per a altres tasques amb un major valor afegit. En aquest projecte s'han utilitzat Xarxes Neuronals de Convolució com a instrument clau en el procés de desenvolupament d'un nou model de classificació dels Talls Biomètrics Fetals Estàndard, actuant com a Extractor de Característiques de les imatges i com a classificador.

El treball realitzat es divideix en dues parts. En primer lloc, l'obtenció del conjunt d'Imatges Ecogràfiques i de la construcció del Conjunt de Dades (conegut en anglès com a DataSet), amb la distinció de tres classes d'imatges segons els plans biomètrics tinguts en compte (BPD, CRL i NT). Una vegada llest el Conjunt de Dades, acte seguit es procedeix a l'aplicació d'un preprocessat automatitzat per tal d'eliminar totes les anotacions, marques de color de la imatge, reducció del soroll i realçar les estructures anatòmiques importants.

La segona part consisteix ja en tot allò referent a les tècniques de classificació, on es treballa amb el Conjunt de Dades creat amb les imatges ja preprocessades. Una vegada arribat a aquest punt, es varen seguir tres metodologies diferents, una on s'utilitzaven les CNN seleccionades com a Extractors de Característiques de les nostres imatges, on una vegada extretes les característiques de tot el conjunt d'imatges, s'utilitzava un SVM com a classificador. Un altre dels mètodes utilitzats consistia en la utilització de les CNN seleccionades, però aplicant diversos canvis que permeten el seu ús per a les característiques del nostre Conjunt de Dades. Per últim, el tercer mètode consisteix en l'elaboració d'una nova CNN a mida per al problema tractat. Durant el desenvolupament del projecte, s'han anat observant els tres específics de cada una de les metodologies utilitzades. S'ha trobat que el primer mètode que aquí hem nombrat és el més eficaç en quant es disposa d'un Conjunt de Dades no molt gran, mentre que els altres dos necessiten una quantitat més gran de dades per poder assolir els mateixos nivells d'eficàcia en la realització de la tasca. El problema derivat d'aquest requisit és el gran augment del cost computacional que s'experimenta. En canvi, entre aquests dos últims mètodes, també s'hi observa una gran diferència, ja que en l'experiment on es dissenya una CNN a mida, al

començar el procés d'aprenentatge, tots els paràmetres de la xarxa s'inicialitzen aleatòriament, mentre que en l'altre es troben inicialitzant amb els valors apresos després d'haver passat per un procés d'aprenentatge amb el Conjunt ImageNet, aquest fet ajuda a que la solució convergeixi cap una millor eficàcia. Com que al final es va disposar d'un Conjunt de Dades suficientment gran, es va comprovar que el segon mètode acabava el procés amb una eficàcia major que els altres.

Finalment s'han obtingut diversos models amb la capacitat de desenvolupar la tasca amb més o menys eficàcia, però amb l'obtenció d'un model basat en la tècnica de Transferència de l'Aprenentatge que permet una classificació dels Talls Biomètrics Fetals amb un nivell d'eficàcia satisfactori.

Vists els resultats assolits, es considera la tasca d'automatització de la classificació dels Talls Biomètrics Fetals acabada. Per tant, el següent pas que caldria realitzar consisteix en l'elaboració d'un programa amb la capacitat de calcular automàticament les mesures anatòmiques importants de cadascun dels Talls Biomètrics Fetals. Per tant, amb això es proveiria als professionals sanitaris d'una eina que resultaria clau en l'augment de l'eficiència de les decisions clíniques derivades dels processos de diagnòstic.

Pressupost

En aquest apartat es procedirà al desglossament i justificació del pressupost. Aquest, consisteix en la valoració econòmica dels software utilitzat, costos d'enginyeria i materials i la mà d'obra.

Costos del Projecte	Tasca	Hores	Preu / hora	Total
Desenvolupament del model	Cerca d'informació	120	40 €	4800 €
	Estat de l'Art	60	40 €	2400 €
Implementació	Preprocessat	10	40 €	400 €
	Condicionament dels models i classificació	350	40 €	14.000 €
Memòria del projecte	Metodologies i bibliografia	110	40 €	4400 €
	Resultats i conclusions	50	40 €	2000 €
Estació de treball				2000 €
Software	Llicència de MATLAB i Llibreries			5000 €
Total				35.000 €

Bibliografia

[1] A. Krizhevsky; I. Sutskever; G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", 2012

[2] A. López, E; Sabrià J.; Arévalo S.; Borrell, A.; Dra. Gómez, T.; Juan, M.; Muñoz, B.; Palau,J.; Torrent,A.; Zientalska, "Secció d'Ecografia i Medicina Fetal de la Societat Catalana d'Obstetrícia i Ginecologia. GUIA DE L 'ECOGRAFIA OBSTÈTRICA DEL PRIMER TRIMESTRE," 2016.

[3] D. Soekhoe; P. Van der Putten; A. Plaat, "On the Impact of data set Size in Transfer Learning using Deep Neural Networks", LIACS Leiden University, 2016

[4] F. Chollet, "Deep Learning with Python", Manning, 2018

[5] GE Healthcare, "Voluson E6 GE Healthcare," 2017. [Online]. Available: http://www3.gehealthcare.com/en/products/categories/ultrasound/voluson/voluson_e6. [Accessed: 7-Maig-2018].

[6] GE Healthcare, "Voluson E8 GE Healthcare," 2017. [Online]. Available: http://www3.gehealthcare.com/en/products/categories/ultrasound/voluson/voluson_e8. [Accessed: 7-Maig-2018].

[7] H. Wu; Z. Deng; B. Zhang; Q. Liu; J. Chen, "Classifier Model Based on Machine Learning Algorithms: Application to Differential Diagnosis of Suspicious Thyroid Nodules via Sonography", American Roentgen Ray Society, 2015

[8] J.Chi; E. Walia; P. Babyn; J. Wang; G. Groot; M. Eramian, "Thyroid Nodule Classification in Ultrasound Images by Fine-Tuning Deep Convolutional Neueal Network", 2017

[9] J. Sang; J. Ok, "Multiclass Probabilistic Classification for Support Vector Machines", IEICE TRANS. INF. & SYST. VOL.E98-D, NO.6 JUNE 2015, 2015

[10] Keras Documentation, "Keras Applications". [Online]. Disponible: https://keras.io/applications/

[11] K. Kaur, "Digital Image Processing in Ultrasound Image", International Journal on Recent and Innovation Trends in Computing and Communication, 2013

[12] Mathworks, “Applying Supervised Learning,” *What is Machine Learning*, 2016. [Online]. Available: https://es.mathworks.com/solutions/machine-learning.html. [Accessed: 7-Maig-2018].

[13] Mathworks, “Deep Learning in MATLAB”, [Online]. Disponible: https://es.mathworks.com/help/nnet/ug/deep-learning-in-matlab.html, 2018

[14] MathWorks, “Pretrained AlexNet convolutional neural network” [Online]. Disponible: https://es.mathworks.com/help/nnet/ref/alexnet.html, 2016.

[15] M. Bennasar; V. Borobio; B. Puerto, “PROTOCOLOS MEDICINA FETAL Y PERINATAL. Servicio de Medicina Maternofetal” 2016

[16] Qing Li ; Weidong Cai ; Xiaogang Wang ; Yun Zhou ; David Dagan Feng ; Mei Chen, “Medical Image Classification with Convolutional Neural Network”, IEEE, 2014

[17] R. Benitez; G. Escudero; S. Kanaan, “Inteligencia Artificial Avanzada”, UOC (Universitat Oberta de Catalunya, 2013

[18] S. P. Imaging, “ACUSON S2000 Ultrasound System,” *Ultrasound*, 2017. [Online]. Available: https://static.healthcare.siemens.com/siemens_hwem-hwem_ssxa_websites-contextroot/wcm/idc/groups/public/@global/@imaging/@ultrasound/documents/download/mdaw/ndax/~edisp/s2000_transducer_2013-00301529.pdf.[Accedit:7-Maig 2018]

[19] S. Bendali, “Biometric Plane Classification in Fetal Ultrasound Scan”, Treball de Final del Grau en Enginyeria Biomèdica, Barcelona, UPC, 2017

Annex A

A1. Codi MATLAB

Script càrrega de les imatges i preprocessat

```
%Funció encarregada de seleccionar la carpeta desitjada i proporcionar
% la seva ruta, les carpetes amb les imatges han de trobar-se aquí
imroot=uigetdir('Select US_RawImageSet100 folder');
D=dir(imroot);% Extracció de les rutes de tot el que hi conté la carpeta
carpetes=length(D);
% Funció per tenir les rutes completes de les carpetes
Dir=nom(D, carpetes);
% Eliminació d'aquelles carpetes que no contenen imatges i d'allò que no
% és una carpeta
j=1;
for i=1:carpetes %Per buidar carpetes sense imatges
imge = imageSet(Dir(i).folder);
    if imge.Count~=0
        imgSet(j)=imge;
        j=j+1;
    end
end
% Especificació de la mida que tindran les imatges
ImShape=299;
% Guardat de les imatges preprocessades per a poder disposar d'elles
% sempre que es vullgui
cd(uigetdir)
l=length(PreP{1,2});
for j = 1:l
    L=PreP{1,2}{j,1};
    Im = cat(3,L,L,L);%Convertim la imatge a format 3 canals
    imwrite(Im,['image' num2str(j) '.jpeg'])
    clear Im
end

cd(uigetdir)
m=length(PreP{2,2});
j=0
for i = 1:(l+m)
    L=PreP{2,2}{j,1};
    Im = cat(3,L,L,L);
    imwrite(Im,['image' num2str(i) '.jpeg'])
    clear Im
    j=j+1
end

cd(uigetdir)
n=length(PreP{3,2});
j=0
for i =(l+m):(l+m+n)
```

```
    L=PreP{3,2}{j,1};
    Im = cat(3,L,L,L);
    imwrite(Im,['Simage' num2str(i) '.jpeg'])
    clear
    j=j+1
end
```

Funció nom.m

```
function [Directoris] = nom(D, carpetes)
% s2='\';
Directoris=D;%En comptes de tenir que reutilitzar la matriu creada per la
         %funció 'dir', crear un vector amb els directoris de les
carpetes
for i=1:carpetes
    Directoris(i).folder = fullfile(D(i).folder, D(i).name);
end
end
```

Funció Imatges.m

```
% En aquesta funció es recibeixen les imatgen en format ruta, es
% converteixen a matriu i es truca a les funcions que li aplicaran el
% preprocessat a cada imatge

function [imatges] = Imatges(imgSet, ImShape)

for i=1:length(imgSet)
    imatges{i,1}=imgSet(1,i).Description;
    n=imgSet(1,i).Count;
    for j=1:1:n
        e=imread(imgSet(1,i).ImageLocation{1,j});
        I{j,1}=e;
        [R, p] = RemoveMarks(e);
        A= ImPrep(R, p, ImShape);
        J{j,1}=A;

    end

    imatges{i,2}=J;
    clear J

end

end
```

Funció RemoveMarks.m

```
function [newIm]= RemoveMarks(RGB)

% Preprocessing 2D Ultrasound Images :
%-------------------------------------

% RGB : input image
% newIm: output image

% RemoveMarks subtasks:
    % 1- Removes irrelevant RGB pixels form the image with smoothing up
    % letters within the ultrasound image to get more uniformity in the
    % image
    % To replace the removed pixels with a more uniform surrounding
    % intensity  we create the following filtering function kernel with
    % size of [20 20] to perform unifor intensity calculation around
    % taregeted pixels with anaverage filter patch of size [20 20]

% Define image I
I=RGB;

% Kernel of average patch
kernel= fspecial('average',[25 25]);


%=======FIND COLOR MARKS AND ANNOTATIONS PIXELS:
% 1- BLUE PIXELS:

% Define thresholds for Blue pixels values based on the histogram image
settings
channel1Min = 0.000;
channel1Max = 143.000;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.000;
channel2Max = 255.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 162.000;
channel3Max = 255.000;

% Find those pixels

t = find((I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max));

% Replace them with the median value of the size grid indicated

% The image I in converted to grayscale to apply filters
G= rgb2gray(I);
grayI= G;
```

```
blurred1= imfilter(grayI(t), kernel);
G(t) = blurred1;

% 2- YELLOW PIXELS
channel1Min = 60.000;
channel1Max = 255.000;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 190.000;
channel2Max = 255.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 189.000;

t = find((I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max));

blurred2= imfilter(grayI(t), kernel);
G(t) = blurred2;

% 3- WHITE PIXELS

channel1Min = 161.000; channel1Max = 255.000;
channel2Min = 255.000; channel2Max = 255.000;
channel3Min = 0.000; channel3Max = 255.000;

t = find((I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max));

blurred3= imfilter(grayI(t), kernel);
G(t) = blurred3;

% 4- SOME GREEN ONES

channel1Min = 0.000; channel1Max = 65.000;
channel2Min = 111.000; channel2Max = 238.000;
channel3Min = 0.000; channel3Max = 86.000;

t = find((I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max));

blurred4= imfilter(grayI(t), kernel);
G(t) = blurred4;

newG = cat(1,G);

end
```

Funció ImPrep.m

```
function [newIm]=ImPrep(Grey, p, ImShape)

% Preprocessing 2D Ultrasound Images :
%-------------------------------------

% RGB  : input image
% newIm: output image

% ImPrep subtasks:
    % 2- Converts the image to grayscale
    % 3- Contrast adjustment
    % 4- Denoising with a median filter
    % 5- Image resizing
newGM=p;
% Define image I
grayI= Grey;

% 2D median filtering kernel
k=[6 6];

% The image I in converted to grayscale to apply median filter
filtered = medfilt2(grayI, k);

adjusted= imadjust (newGM);

% New resulting image:
J=ImShape+30;
I =  imresize(filtered, [J ImShape]);
newIm=I(31:J,:,:);
end
```

Script utilització AlexNet com a Extractor de Característiques i un SVM com a classificador

```
images = imageDatastore('DataSet',...
    'IncludeSubfolders',true,...
    'LabelSource','foldernames');

% Partició del conjunt de dades en 80% entrenament i 20% validació
[trainingImages,testImages] = splitEachLabel(images,0.8, 0.2, ...
    'randomized');

%  Display some sample images
numTrainImages = numel(trainingImages.Labels);
idx = randperm(numTrainImages,16);
figure
for i = 1:16
```

```
    subplot(4,4,i)
    I = readimage(trainingImages,idx(i));
    imshow(I)
end

%%
% Càrrega de la xarxa preentrenada
net = alexnet;

% Selecciona fins quina capa de la xarxa vols treballar, i extrau les
% característiques de les imatges d'entrenament i validació
layer = 'fc7';
trainingFeatures = activations(net,trainingImages,layer);
testFeatures = activations(net,testImages,layer);

% Extracció de les etiquetes de cada classe per al conjunt d'entrenament
i
% per al de validació
trainingLabels = trainingImages.Labels;
testLabels = testImages.Labels;

% Utilitza les caracteristiques extretes del conjunt d'entrenament per
% entrenar el classificador (SVM)
classifier = fitcecoc(trainingFeatures,trainingLabels);

%%
% Classify Test Images
% Utilització del model SVM entrenat per a la predicció de les classes de
%les mostres de validació
predictedLabels = predict(classifier,testFeatures);

% Comprovació visual de si un petit conjunt de les imatges de validació
% s'han classificat correctament
idx = [1 5 10 15];
figure
for i = 1:numel(idx)
    subplot(2,2,i)
    I = readimage(testImages,idx(i));
    label = predictedLabels(idx(i));
    imshow(I)
    title(char(label))
end

% Calcular la pressició en la classificació del classificador
accuracy = mean(predictedLabels == testLabels)
```

Script utilització AlexNet amb la tècnica Transferència de l'Aprenentatge

```
images = imageDatastore('ImatgesTotal',...
    'IncludeSubfolders',true,...
    'LabelSource','foldernames');

% Partició del conjunt de dades en 80% entrenament i 20% validació
[trainingImages,validationImages] = splitEachLabel(images,0.8, 0.2, ...
    'randomized');

%  Display some sample images
numTrainImages = numel(trainingImages.Labels);
idx = randperm(numTrainImages,16);
figure
for i = 1:16
    subplot(4,4,i)
    I = readimage(trainingImages,idx(i));
    imshow(I)
end


%%
% Càrrega de la xarxa preentrenada
net = alexnet;

% Extracció de totes les capes exceptuant les últimes tres, que es troben
% configurades per a un problema amb imatges de 1000 classes diferents
layersTransfer = net.Layers(1:end-3);

% Transferència de les capes amb els seus corresponents pesos al nou
%model que realitzarà la tasca de classificació, i incorporació dels tres
%últimes capes encarregades de permetre la classificació entre 3 classes
%d'imatges
numClasses = numel(categories(trainingImages.Labels))
layers = [
    layersTransfer
    fullyConnectedLayer(numClasses,'WeightLearnRateFactor',20, ...
        'BiasLearnRateFactor',20)
    softmaxLayer
    classificationLayer];

% Especificació de les opcions d'entrenament
miniBatchSize = 30;
numIterationsPerEpoch =
floor(numel(trainingImages.Labels)/miniBatchSize);
options = trainingOptions('sgdm',...
    'MiniBatchSize',miniBatchSize,...
    'MaxEpochs',4,...
    'InitialLearnRate',1e-4,...
    'Verbose',false,...
    'Plots','training-progress',...
    'ValidationData',validationImages,...
    'ValidationFrequency',numIterationsPerEpoch, ...
```

```
    'ExecutionEnvironment','cpu' );

% Entrena la nova CNN, que consisteix en les capes transferides i les
%noves afegides
netTransfer = trainNetwork(trainingImages,layers,options);
[YPred,scores] = classify(netTransfer,validationImages);
```

A2. Codi Python

Script utilitzant VGG19 com a Extractor de Característiques i SVM com a classificador

```
import numpy
import os
import skimage
import random
import cv2
from skimage import data
from PIL import Image
from keras.utils import np_utils

x=os.listdir('.')
print(x)
directori=os.getcwd()+'/'+x[i]
print(directori)
y=os.listdir(directori)
print(y)
directori1=directori+'/'+y[0]
y1=os.listdir(directori1)
directori2=directori+'/'+y[1]
y2=os.listdir(directori2)
directori3=directori+'/'+y[2]
y3=os.listdir(directori3)

random.shuffle(y1)
random.shuffle(y2)
random.shuffle(y3)


n=506
p1= ['']*n
p2= ['']*n
p3= ['']*n
for i in range(0, n):
    p1[i]=directori1+'/'+y1[i]
    p2[i]=directori2+'/'+y2[i]
    p3[i]=directori3+'/'+y3[i]



trainingImagesD=['']*(406*3)
```

```
testImagesD=['']*(100*3)
trainingLabels=numpy.empty(406*3)
testLabels=numpy.empty(100*3)

j=0
for i in range(0, 406):
    trainingImagesD[j]=p1[i]
    trainingLabels[j]=0
    j=j+1

for i in range(0, 406):
    trainingImagesD[j]=p2[i]
    trainingLabels[j]=1
    j=j+1

for i in range(0, 406):
    trainingImagesD[j]=p3[i]
    trainingLabels[j]=2
    j=j+1

j=0
for i in range(406, 506):
    testImagesD[j]=p1[i]
    testLabels[j]=0
    j=j+1

for i in range(406, 506):
    testImagesD[j]=p2[i]
    testLabels[j]=1
    j=j+1

for i in range(406, 506):
    testImagesD[j]=p3[i]
    testLabels[j]=2
    j=j+1


tI=numpy.arange((227*227*3)).reshape((227,227,3))
trainingImages=numpy.arange((1218*224*224*3)).reshape((1218,224,224,3))
for i in range(0, 1218):
    tI=skimage.data.imread(trainingImagesD[i], as_grey=False)
    for j in range(0, 3):
        for k in range(0, 224):
            for l in range(0, 224):
                trainingImages[i, k, l, j]=tI[k, l, j]
```

```
tstI=numpy.arange((227*227*3)).reshape((227,227,3))
testImages=numpy.arange((300*224*224*3)).reshape((300,224,224,3))
for i in range(0, 300):
    tstI=skimage.data.imread(testImagesD[i], as_grey=False)
    for j in range(0, 3):
        for k in range(0, 224):
            for l in range(0, 224):
                testImages[i, k, l, j]=tstI[k, l, j]
```

```
from keras.applications import VGG19
from keras.models import Model
base_model = VGG19(weights='imagenet')
model = Model(inputs=base_model.input,
outputs=base_model.get_layer('fc2').output)

trainingFeatures = model.predict(trainingImages)
testFeatures = model.predict(testImages)
```

```
from sklearn import svm
from sklearn.metrics import accuracy_score, confusion_matrix

svc = SVC(C=100, kernel='poly', degree=2)
svc = svc.fit(trainingFeatures, trainingLabels)
solu=svc.predict(testFeatures)

print(accuracy_score(testLabels, solu))
print(confusion_matrix(testLabels, solu))
```

Script utilitzant InceptionV3 com a Extractor de Característiques i SVM com a classificador

```
import os
import random
import numpy
import skimage
from skimage import data
from keras.utils import np_utils


x=os.listdir('.')
print(x)
directori=os.getcwd()+'/'+x[8]
print(directori)
y=os.listdir(directori)
print(y)
directori1=directori+'/'+y[0]
y1=os.listdir(directori1)
directori2=directori+'/'+y[1]
y2=os.listdir(directori2)
directori3=directori+'/'+y[2]
y3=os.listdir(directori3)

random.shuffle(y1)
random.shuffle(y2)
random.shuffle(y3)


n=506
p1= ['']*n
p2= ['']*n
p3= ['']*n
for i in range(0, n):
    p1[i]=directori1+'/'+y1[i]
    p2[i]=directori2+'/'+y2[i]
    p3[i]=directori3+'/'+y3[i]


trainingImagesD=['']*(406*3)
testImagesD=['']*(100*3)
trainingLabels=numpy.empty(406*3)
testLabels=numpy.empty((100*3))

j=0
```

```
for i in range(0, 406):
    trainingImagesD[j]=p1[i]
    trainingLabels[j]=0
    j=j+1

for i in range(0, 406):
    trainingImagesD[j]=p2[i]
    trainingLabels[j]=1
    j=j+1

for i in range(0, 406):
    trainingImagesD[j]=p3[i]
    trainingLabels[j]=2
    j=j+1

j=0
for i in range(406, 506):
    testImagesD[j]=p1[i]
    testLabels[j]=0
    j=j+1

for i in range(406, 506):
    testImagesD[j]=p2[i]
    testLabels[j]=1
    j=j+1

for i in range(406, 506):
    testImagesD[j]=p3[i]
    testLabels[j]=2
    j=j+1

print(testImagesD)
trainingImages=numpy.arange((1218*299*299*3)).reshape((1218,299,299,3))
for i in range(0, 1218):
    trainingImages[i]=skimage.data.imread(trainingImagesD[i],
as_grey=False)

testImages=numpy.arange((300*299*299*3)).reshape((300,299,299,3))
for i in range(0, 300):
    testImages[i]=skimage.data.imread(testImagesD[i], as_grey=False)



trainingImagesIncep = trainingImages/ 255
testImagesIncep = testImages / 255
```

```
from keras.applications.inception_v3 import InceptionV3
from keras.models import Model
base_model = InceptionV3(weights='imagenet')
model = Model(inputs=base_model.input,
outputs=base_model.get_layer('avg_pool').output)


trainingFeatures = model.predict(trainingImages)
testFeatures = model.predict(testImages)


from sklearn import svm
from sklearn.metrics import accuracy_score, confusion_matrix

svc = SVC(C=100, kernel='poly', degree=2)
svc = svc.fit(trainingFeatures, trainingLabels)
solu=svc.predict(testFeatures)

print(accuracy_score(testLabels, solu))
print(confusion_matrix(testLabels, solu))
```

Script utilitzant VGG19 amb Transferència de l'Aprenentatge

```
from keras.applications.vgg19 import VGG19
from keras.preprocessing import image
from keras.applications.vgg19 import preprocess_input
from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D


base_model = VGG19(weights='imagenet')
model = Model(inputs=base_model.input,
outputs=base_model.get_layer('fc2').output)
x = model.output
predictions = Dense(3, activation='softmax')(x)
NouModel = Model(inputs=model.input, outputs=predictions)

for layer in model.layers:
    layer.trainable = False

model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
              metrics=['accuracy'])
history = model.fit(trainingImages, trainingLabels,
                    validation_data=(testImages, testLabels), epochs=10,
verbose=2)


NouModel.summary()


trainingLabels = np_utils.to_categorical(trainingLabels)
testLabels = np_utils.to_categorical(testLabels)


model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
metrics=['accuracy'])
history = model.fit(trainingImages, trainingLabels,
validation_data=(testImages, testLabels), epochs=10,  verbose=2)

import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()


del history

for layer in model.layers[:17]:
   layer.trainable = False
for layer in model.layers[17:]:
   layer.trainable = True

from keras.optimizers import SGD
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9),
              loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(trainingImages, trainingLabels,
                    validation_data=(testImages, testLabels), epochs=10,
verbose=2)
```

Script utilitzant InceptionV3 amb Transferència de l'Aprenentatge

```
from keras.applications.inception_v3 import InceptionV3
from keras.preprocessing import image
from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D
from keras import backend as K

base_model = InceptionV3(weights='imagenet', include_top=False)
base_model.summary()
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(3, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)

for layer in base_model.layers:
    layer.trainable = False

model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
metrics=['accuracy'])
history = model.fit(trainingImages, trainingLabels, epochs=10,
verbose=2)

trainingLabels = np_utils.to_categorical(trainingLabels)
testLabels = np_utils.to_categorical(testLabels)

trainingImages = trainingImages / 255
testImages = testImages / 255

for layer in model.layers[:249]:
   layer.trainable = False
for layer in model.layers[249:]:
   layer.trainable = True

from keras.optimizers import SGD
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9),
              loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(trainingImages, trainingLabels,
                    validation_data=(testImages, testLabels), epochs=4,
verbose=2)

resultats=model.predict(testImages)
```

Script disseny de la CNN a mida

```
from keras import layers
from keras import models
import numpy
import os
import sklearn.preprocessing
import skimage
import random
import cv2
from skimage import data
from PIL import Image
from keras.utils import np_utils
from keras import backend as K
from keras import optimizers


model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(227,
227, 3)))
model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(600, activation='relu'))
model.add(layers.Dense(3, activation='softmax'))

model.summary()


x=os.listdir('.')
print(x)
directori=os.getcwd()+'/'+x[11]
print(directori)
y=os.listdir(directori)
print(y)
directori1=directori+'/'+y[0]
```

```
y1=os.listdir(directori1)
directori2=directori+'/'+y[1]
y2=os.listdir(directori2)
directori3=directori+'/'+y[2]
y3=os.listdir(directori3)

random.shuffle(y1)
random.shuffle(y2)
random.shuffle(y3)


n=506
p1= ['']*n
p2= ['']*n
p3= ['']*n
for i in range(0, n):
    p1[i]=directori1+'/'+y1[i]
    p2[i]=directori2+'/'+y2[i]
    p3[i]=directori3+'/'+y3[i]


trainingImagesD=['']*(406*3)
testImagesD=['']*(100*3)
trainingLabels=numpy.empty(406*3)
testLabels=numpy.empty((100*3))

j=0
for i in range(0, 406):
    trainingImagesD[j]=p1[i]
    trainingLabels[j]=0
    j=j+1

for i in range(0, 406):
    trainingImagesD[j]=p2[i]
    trainingLabels[j]=1
    j=j+1

for i in range(0, 406):
    trainingImagesD[j]=p3[i]
    trainingLabels[j]=2
    j=j+1

j=0
for i in range(406, 506):
    testImagesD[j]=p1[i]
    testLabels[j]=0
```

```
    j=j+1

for i in range(406, 506):
    testImagesD[j]=p2[i]
    testLabels[j]=1
    j=j+1

for i in range(406, 506):
    testImagesD[j]=p3[i]
    testLabels[j]=2
    j=j+1


print('Standaritzant')
trainingImages=numpy.arange((1218*227*227)).reshape((1218,227,227))
for i in range(0, 1218):
    trainingImages[i]=skimage.data.imread(trainingImagesD[i])

testImages=numpy.arange((300*227*227)).reshape((300,227,227))
for i in range(0, 300):
    testImages[i]=skimage.data.imread(testImagesD[i])

trainingImages = trainingImages / 255
testImages = testImages / 255

trainingLabels = np_utils.to_categorical(trainingLabels)
testLabels = np_utils.to_categorical(testLabels)


model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
metrics=['accuracy'])
history = model.fit(trainingImages, trainingLabels,
validation_data=(testImages, testLabels), epochs=20, verbose=2)
resultats=model.predict(testImages)


from sklearn.metrics import confusion_matrix, accuracy_score
accuracy_score(testLabels, resultats)
confusion_matrix(testLabels, resultats)
```